VŠB - Technická univerzita Ostrava
Fakulta Elektrotechniky a informatiky
Katedra informatiky

Mobilní navigace v prostředí DirectX

Mobile Navigation Using Mobile DirectX Platform

Ostrava 2010 Bc. Pavel Dvouletý

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Pavel Dvouletý**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Mobilní navigace v prostředí DirectX
Mobile Navigation Using Mobile DirectX Platform

Zásady pro vypracování:

Tato diplomová práce si klade za cíl zmapovat současný stav technologií vykreslování 3D scény na mobilních zařízeních s Windows Mobile 5 a Windows Mobile 6 Standard (Direct3D, DirectDraw, DirectShow a OpenGL ES). Cílem je vytvořit systém umožňující 3D navigaci v definovaném prostoru (např. archeologické naleziště, ZOO apod.) Aplikace běžící na mobilním zařízení bude schopna zobrazit 3D scénu, tak jak ji reálně vidí uživatel systému a zobrazit k dané lokalitě detailní informace. Poloha zařízení se určí využitím zabudovaného GPS přijímače.

1. Aplikace pro mobilní zařízení.
2. Aplikace (desktopová) pro tvorbu 3D scény ve formátu vhodného pro mobilní zařízení.
3. Ukázka funkčnosti (např. areál Ostravské ZOO).
4. Testování aplikace v terénu.
5. Shrnutí dosažených výsledků.

Seznam doporučené odborné literatury:

http://archeoguide.intranet.gr/papers/publications/ARCHEOGUIDE-BIS02.pdf
http://ieeexplore.ieee.org/iel5/7966/22019/01024080.pdf
http://msdn.microsoft.com/en-us/library/aa452478.aspx

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Michal Krumnikl**

Datum zadání: 30.11.2008

Datum odevzdání: 07.05.2010

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr.Ing. Eduard Sojka
vedoucí katedry

prof. Ing. Ivo Vondrák, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 7. 5. 2010 ……..

Poděkování

Za vedení a pomoc při tvorbě této diplomové práce bych chtěl poděkovat Ing. Michalu Krumniklovi. Dále bych chtěl poděkovat rodině, přítelkyni a také svým kamarádům za morální podporu a pomoc.

Abstrakt a klíčová slova

Hlavním cílem této práce je prozkoumání možností tvorby a využití 3D aplikací na platformě Windows Mobile 5 a 6. Úkolem je vytvořit funkční GPS navigaci na 3D mapě zvolené lokality pro mobilní zařízení s tímto operačním systémem. Mimo 3D navigace po zvolené lokalitě bude aplikace obsahovat informace jak textové, tak i obrazové o jednotlivých částech lokality. 3D model by měl věrně reprezentovat lokalitu, aby uživatelům ulehčoval vizuální koordinaci mezi navigací a skutečností.

Klíčová slova: Windows Mobile, DirectX Mobile, GPS, 3D, Blender

Abstract and key words

The main purpose of this thesis is to explore the possibilities and usage of 3D applications on the Windows Mobile 5 and 6 platforms. The main task is to develop an application with GPS navigation on a 3D map of chosen locality for this platform. The application will provide additional information (text and pictures) for each part of the locality. The 3D model should be an accurate representation of the locality as it aims to help the user to coordinate between reality and a 3D map.

Key words: Windows Mobile, DirectX Mobile, GPS, 3D, Blender

Seznam použitých symbolů a zkratek

API – Application Programming Interface
CSS - Cascading Style Sheets
GNU – General Public License
GPL – General Public Licence
GPS – Global Positioning System
HAL – Hardware Abstraction Layer
HEL – Hardware Emulation Layer
HTML – Hyper Text Markup Language
Java ME – Java Micro Edition
Java SE – Java Standard Edition
M3G – Mobile 3D Graphics
OS – Operační systém
PDA – Personal Digital Assistant
S-JTSK – Souřadnicový systém jednotné trigonometrické sítě katastrální
SDK – Software Development Kit
URL – Uniform Resource Locator
UTM – Universal Transverse Mercator
W3C – World Wide Web Consortium
WGS – World Geodetic System
WMS – Web Map Service
XML – Extensible Markup Language

Obsah

Přílohy

A. Výpis z SVG souboru s cestami
B. Textura modelu
C. Vzorový vstupní XML soubor
D. Příkladový detail atrakce - HTML kód
E. Obsah přiloženého CD

Úvod

Pokrok a zdokonalování mobilních zařízení probíhá ve velkém tempu. Dnešní zařízení svým výkonem a výbavou mnohokrát převyšují zařízení, která byla na trhu před deseti lety. Z tohoto důvodu je na vývojáře software pro tyto zařízení vyvíjen velký tlak. Dochází často k inovacím vývojových nástrojů nebo vydání úplně nových. S rostoucím výkonem zařízení se programátorům dostává více prostředků pro realizaci náročných aplikací. Do této skupiny patří také aplikace využívající 3D grafiky.
Tato diplomová práce klade důraz na prozkoumání možností tvorby 3D aplikace pro mobilní zařízení využívající globální navigace GPS. Zařízení bude používat na platformě Windows Mobile 6 a využije zabudovaného GSP přijímače.
Vývoj takové aplikace se dá rozdělit do tří částí:

- GPS – Prvním krokem je získání GPS dat, jejich zpracovaní, tak aby je bylo dále možno využít pro tvorbu 3D modelu. Zde se budu snažit o prozkoumání a zdokumentování jednotlivých metod, a ty nejvhodnější využít jako podklad pro tvorbu 3D modelu.
- Tvorba 3D modelu – K tvorbě modelu lze využít některé z mnoha aplikací dostupných na trhu. Vybral jsem si aplikaci Blender, která nabízí funkce profesionálních nástrojů, ale podléhá GPL licenci.
- Tvorba aplikace – Návrh a implementace aplikace využije knihovny DirectX Mobile a .NET Compact Framework. K implementaci bude využito vývojového prostředí Visual Studio .NET 2008.

Cílem této práce je zmapovat možnosti při vývoji 3D aplikace využívající GPS navigace, vytvořit 3D model reálné lokality a vyprodukovat funkční aplikaci, která poslouží jako demonstrace. Největší důraz je kladen na vývoj aplikace. Vytvořený 3D model bude zobrazovat pouze malé území a jeho hlavním účelem je využití při vývoji a testování aplikace. U aplikace budu hodnotit využitelnost knihovny DirectX Mobile a celkový chod aplikace na fyzickém zařízení.
Prvních pět kapitol je věnováno popisu využitých technologii, vysvětlení základních pojmů a postupů. Počínaje šestou kapitolou začíná popis postupu, který byl použit k získání GPS dat, vytvoření 3D modelu, návrhu a implementace aplikace. Na závěr jsou shrnuty výsledky testování.

1. Operační systémy mobilních zařízení

V první kapitole uvádím krátký přehled operačních systémů (dále jen OS) pro mobilní zařízení. Tato diplomová práce se soustředí pouze na Windows Mobile OS, který je proto popsán podrobněji. U ostatních je uváděna krátká charakteristika OS, seznam programovacích jazyků, které lze využít a podporované knihovny pro vytváření 3D aplikací. V této kapitole se budu také zmiňovat o třech knihovnách a to M3G[1] (Mobile 3D Graphics API, založeno na jazyce Java), OpenGL ES[1] a DirectX Mobile[1].

1.1. Konkurenti Windows Mobile

Níže uvedení konkurenti Windows Mobile patří mezi ty nejužívanější OS na mobilních zařízení. Každý z těchto OS je vyvíjen jinou společností.

1.1.1. Symbian OS

Operační systém vyvíjený společností Symbian Software pod záštitou společnosti Nokia. Množství zařízení na trhu používajících Symbian OS se pohybuje okolo čtyřiceti procent. Celý systém je naprogramován v jazyce C++. Na úplném počátku se systém jmenoval EPOC a jeho vývoj se datuje od počátku devadesátých let. První operační systém pod názvem Symbian byl představen roku 2001 s počátečním číslem verze 6.0 až po dnešní verzi 9.5. Symbian OS využívá mnoho výrobců mobilních zařízení (např. Nokia, SonyEricsson, Siemens, Samsung).
Hlavním programovacím jazykem pro Symbian je C++. Pro vývoj software lze dále využít jazyků Java Me(Java Micro Edition), Flash Lite, C a Ruby. Použití C# a Visual Basic pod .NET Framework je možné při použití přídavného modulu AppForge. Je zde možnost využít grafických knihoven M3G a OpenGL ES.

1.1.2. OS X

OS je uváděn také pod názvy iPhone OS nebo OS X iPhone. OS X je využíván pouze jediným zařízením, a to iPhone. OS X i mobilní zařízení vytvořila společnost Apple. První předvedení na trh proběhlo roku 2008 s verzí 1.0. Verze 3.2 byla představena na počátku roku 2010. Celý OS X byl naprogramován v jazyce C.
K vývoji software se využívá iPhone SDK. Podpora programovacích jazyků je následující: C, C++, Objective-C nebo JavaScript. OS X přímo podporuje knihovnu OpenGL ES. Nevýhodou OS X je, že nenabízí podporu jazyka Java ME. Knihovnu M3G tak nelze využít.

[1] Popis knihoven M3G, OpenGL a DirectX Mobile viz. kapitola 2 Knihovny pro vývoj 3D aplikací

1.1.3. BlackBerry OS

OS používaný na zařízeních s názvem BlackBerry vyvíjených kanadskou společností RIM (Research In Motion). Jeho první verze vznikla roku 1999. Poslední verze nese označení 5.0 a byla uvolněna roku 2010. BlackBerry OS je naprogramován v jazyce C++.
Pro vývoj software se využívá výhradně jazyka Java ME (M3G) a jazyky C a C++ nejsou vůbec podporovány. Tento OS však nabízí podporu také pro knihovnu OpenGL ES pomocí API JSR 239, který využívá jazyka Java.

1.1.4. Android

Nejnovější ze všech zde zmíněných OS. Založený na modifikovaném linuxovém jádře společností Android Inc., později zakoupenou firmou Google. Jeho vývoj je rapidní a do budoucna se počítá, že bude hrát na trhu mobilních zařízení důležitou roli. Od počátečního nasazení systému roku 2008 byly vypuštěny updaty, které opravují chyby, optimalizují systém a rozšiřují ho. Poslední z nich nese označení 2.1 (Eclair).
Android nabízí vlastní SDK obsahující vše potřebné k vývoji software. Obsažené knihovny jsou napsány v jazycích C/C++ a obsahují také podporu knihovny OpenGL ES. Další možností je využít jazyka Java (Java ME a M3G).

1.2. Windows Mobile

Jedná se o kompaktní OS určený pro Smartphone a PDA přístroje. Design a funkce jsou podobné desktopové verzi. Windows Mobile je naprogramován v jazyce C++. V dnešní době je na světovém trhu čtvrtým nejoblíbenějším operačním systémem pro mobilní zařízení (první místo patří Symbian OS, následován BlackBerry OS a iPhone OS). Na počátku byl Windows Mobile designován pouze pro kapesní počítače bez možností a funkcí mobilních telefonů. Proto došlo k rozdělení Windows Mobile do několika typů podle cílových zařízení. Nejlépe lze provést demonstraci na verzi Windows Mobile číslo 6, kdy se dodávaly tři variace systému. Windows Mobile Classic, který byl určen zařízením bez možnosti mobilního telefonu, dále Standard, jehož omezení bylo na mobilní telefony (smartphones) bez dotykového displeje. Poslední kategorií je Professional, jedná se o variantu verze 6, která má nejvyšší nároky na mobilní zařízení, včetně dotykového displeje.
Výčet vydaných verzí OS Windows pro mobilní zařízení: Pocket PC 2000, Pocket PC 2002, Windows Mobile 2003, Windows Mobile 5, Windows Mobile 6, Windows Mobile 6.1, Windows Mobile 6.5, Windows Mobile 7.

Součásti Windows Mobile

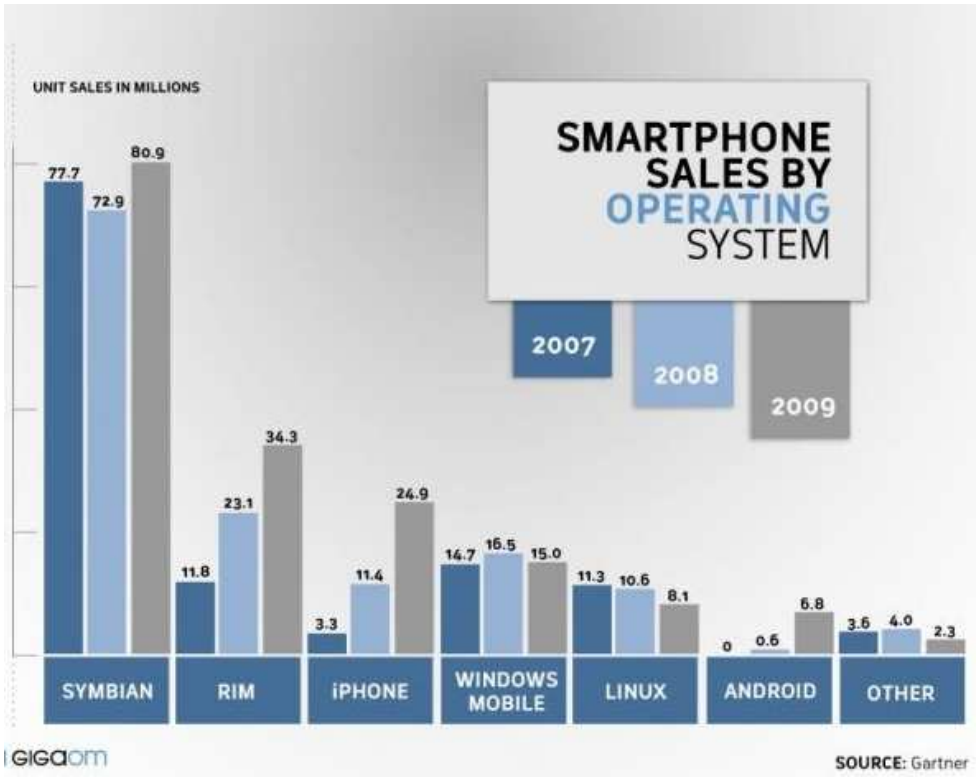
Záměrně zde nezmiňuji popis jednotlivých verzí, protože to není tématem této diplomové práce. Proto tedy uvedu jen krátký popis součástí Windows Mobile.
Od první verze jsou jeho součástí aplikace Pocket Office (Word, Excel, Outlook), Pocket Internet Explorer, Media Player, které se s novými verzemi stale zlepšovaly. Postupně pak byly přidány další aplikace a rozšíření. Mezi ty důležité patří například MSN messanger,

podpora pro DirectShow, GPS interface, ActiveSynch (synchronizace se stolním PC) a mnoho dalších.

Vývoj software

OS Windows Mobile má velice dobré zázemí pro vývoj software. Ke každé verzi systému byla také vydaná SDK. K vývoji aplikací se využívá knihovny .NET Compact Framework[2]. Jako programovací jazyk lze zvolit jazyk C, C++, C# nebo VisualBasic. Pro tento OS je samozřejmostí také podpora jazyka Java, tak jako je tomu u verze pro stolní počítače. OS Windows Mobile podporuje všechny zde zmíněné knihovny (Java 3D, OpenGL ES, DirectX Mobile) pro tvorbu 3D aplikací.

Velké množství informací ke všem výše zmíněným OS leze nalézt na stránkách daného výrobce nebo distributora. Každý z nich má své výhody a nevýhody. Ve své bakalářské práci[3] jsem se zabýval možnostmi platformy Java ME pro vývoj kancelářské aplikace. Rozhodl jsem se proto zdokonalit své znalosti ve vývoji aplikací pro mobilní zařízení s použitím knihoven (.NET Compact Framework a DirectX Mobile) od společnosti Microsoft. Zastoupení Windows Mobile na trhu však není moc výrazné a stále se zhoršuje. Obr. 1.2 zobrazuje graf[4] zastoupení trhu mobilních zařízení vzhledem k použití OS



Obr. 1.2. – Zastoupení trhu s OS

[2] Popis knihovny viz. kapitola 6.1 .NET Compact Framework

[3] Bakalářská práce – Pavel Dvouletý - Osobní organizér pro mobilní zařízeni (2007)

[4] URL: http://gigaom.com/2010/03/18/the-mobile-os-market

2. Knihovny pro vývoj 3D aplikací

Tato kapitola je věnována knihovnám pro vývoj 3D aplikací pro mobilní zařízení. Popis obsahuje základní informace o knihovnách, jejich stručný popis, vlastnosti a použití.

2.1. M3G

Mobile 3D Graphics API definuje balíček pro tvorbu 3D aplikací s využitím programovacího jazyka Java. Rozšiřuje možnosti balíčku Java ME a je určen výhradně pro mobilní zařízení. Nelze jej zaměňovat s balíčkem Java 3D, který rozšiřuje možností balíčku Java SE (Java Standart Edition) a je určen pro desktopové počítače. Balíček byl vyvinut komunitou Java Community Process a jeho první verze nese označení JSR 184. Tato verze byla vydaná roku 2007 s označením 1.1. Nejnovější je verze 2.0, která nese označení JSR 297. Balíček obsahuje přibližně 30 tříd.
M3G je vysokoúrovňovým i nízkoúrovňovým API. Vysokoúrovňová implementace se nazývá "retained" mode a umožňuje rychlou a snadnou tvorbu 3D aplikací. Nevýhodou je náročnost aplikace na CPU. Na některých mobilních zařízeních proto tyto aplikace běží pomalu. Nízkoúrovňová implementace "immediate" mode umožní přímý přístup k vykreslování objektů. Obě zmíněné implementace se dají navzájem kombinovat dle potřeb programátora. Immediate mode využívá implementace OpenGL ES, která obstarává komunikaci mezi hardwarem a příslušnou aplikací. Větší množství informací o tomto balíčku lze nalézt na stránkách Sun Developer System[5].

2.2. OpenGL ES

Open Graphics Library for Embedded Systems je multiplatformní API, které zapouzdřuje tvorbu aplikací 2D nebo 3D grafiky. API se nevyužívá pouze na mobilních zařízeních, ale také na herních konzolách. Správu zajišťuje nezisková organizace Khronos Group, Inc. Na vývoji se podílí velké množství firem (Nokia, Sun, NVidia, Ericsson a další). Toto API určené mobilním zařízením je založeno na desktopové verzi OpenGL, ze které byly odstraněny přebytečné a náročné funkce. Snahou Khronos je udržet zpětnou kompatibilitu s desktopovou verzí. Mobilní verze i přesto obsahuje části, které byly speciálně navrženy pro účely těchto zařízení. První verze 1.0 implementovala pouze základní funkce pro tvorbu 3D aplikací. Nejnovější verze má označení 2.0 a není zpětně kompatibilní.
Toto API patří do skupiny nízkoúrovňových a nabízí velice dobré zázemí pro tvorbu 3D aplikací. K vývoji lze využít téměř libovolného programovacího jazyka. Existuje několik implementací OpenGL API. Například implementace The Open Toolkit Library se zaměřuje na platformy Windows a Linux a lze využít těchto programovacích jazyků: C#, VB.net, C++ a další.

[5] Informace o knihovně M3G, URL: http://developers.sun.com/mobility/apis/articles/3dgraphics/

Mnoho informací doplňující informací a tutoriálu lze nalézt na stránkách organizace Khronos Group[6]. Velmi povedený tutoriál pro tvorbu aplikací běžících na OS Windows Mobile s využitím implementace Vincent Mobile 3D Rendering Library se nachází na internetu[7].

2.3. DirectX

API DirectX Mobile je založeno na desktopové verzi DirectX. Uvedu zde proto nejdříve zběžné informace o tomto API.

2.3.1. DirectX

Název DirectX zastřešuje několik API pro tvorbu multimédií, obzvláště pak her a videa na platformě OS Windows. Jednotlivá API jsou označena slovem Direct a dále příponou popisující dané API (DirectMusic, DirectInput, atd.). Proto vznikl název DirectX, kde písmeno X zastupuje všechna obsažena API. První verze DirectX (verze 1.0) nebyla příliš vydařeným API. Programátorova práce byla komplikovaná a knihovny neposkytovaly dobré zázemí k vytváření moderních 3D aplikací. Vše se ale změnilo a v dnešní době (nejnovější verze je 11.0) patří DirectX společně s OpenGL na špici mezi knihovnami pro vývoj 3D software. DirectX je nejvíce využívaný na poli počítačových her. Používá ho však i herní konzole Xbox.
Výčet nejdůležitějších API a jejich krátký popis:

- **DirectDraw** – Zapouzdřuje vše potřebné k tvorbě 2D grafiky. Microsoft nově představil.
- **Direct2D** – Jde vlastně o vylepšenou verzi DirectDraw.
- **Direct3D** – Někdy taky značeno jako D3D. Zabývá se tvorbou aplikací s 3D grafikou.
- **DirectInput** – Slouží k obsluze vstupu z klávesnice, myši a dalších herních vstupních. zařízení (joystick, gamepady, atd.).
- **DirectSound** - Obsluha zvukového zařízení.
- **DirectPlay** – Obsahuje komunikaci po síti, internetu či modemu.
- **DirectShow** – Podpora multimedií (formáty AVI a MPG). Poprvé ve verzi 8.0

DirectX je součásti komponentního modelu COM, který zajišťuje volnost volby programovacího jazyka (Visual C++, Delphi, C# a Visual Basic). Na počátku DirectX bylo všechno programováno v jazyce C. Postupně se přešlo na C++. V dnešní době se začíná využívat jazyka C#. Někteří však neustále využívají jazyka C++. Důvodem toho je, že knihovny jazyka C++ využívají přímého přístupu k systémovým prostředkům. Šetří se tak výpočetním výkonem zařízení a běh aplikace je svižnější.
Architektura DirectX je postavená na dvou implementacích. HAL (Hardware Abstraction Layer), který přímo využívá hardwarových dispozic zařízení, kdežto HEL (Hardware

[6] Stránky věnované OpenGL ES http://www.khronos.org/opengles/
[7] Stránky věnované OpenGL ES pro platformu Windows Mobile
http://www.zeuscmd.com/tutorials/opengles/02-SettingUpYourEnvironment.php

Emulation Layer) implementuje svojí vlastní funkcionalitu. Při zavádění DirectX se provede kontrola systému a na potřebné operace, které systém hardwarově nepodporuje, se použije HEL, zbytek zajišťuje HAL. Toto zajišťuje velkou flexibilitu.

2.3.2. DirectX Mobile

Jedná se o verzi, která je určena pro mobilní zařízení. Došlo k značnému ořezání základního DirectX a importu pouze Direct3D a DirectDraw API. Nejedná se plnohodnotné zástupce těchto dvou API, tak jak jsou obsaženy v desktopové verzi. Některé funkce zůstaly beze změny, u některých došlo k malým změnám a některé byly vypuštěny. Svou strukturou má mobilní DirectX nejblíže desktopové verzi číslo 8.0. Při vývoji aplikace využívající pouze 2D grafiky je výhodné použít DirectDraw namísto Direct3D. Obsahuje množství funkcí pro práci s 2D grafikou s minimálními nároky na hardware. U vývoje 3D aplikace je nutnost využít Direct3D, protože DirectDraw nepodporuje žádné funkce pro vytváření 3D grafiky v aplikacích.

3. Navigace

Pro navigování uživatele se bude využívat zabudovaného GPS přijímače. Tato kapitola se zabývá vším potřebným okolo GPS systému. Dále budou uvedeny možné systémy promítaní, ve kterých se mapy dají zobrazovat. V poslední částí je zmínka o způsobech, jakými lze získat GPS data, tak aby je bylo možné následně zpracovat a vytvořit mapu ve 3D prostoru.

3.1. GPS

GPS (Global Positioning System) je globální satelitní navigace zajišťující lokalizaci a navigaci za každého počasí a času kdekoli na Zemi. Tento systém byl původně určen pro vojenské účely a je majetkem Spojených států amerických. Vypouštění družic do vesmíru probíhalo v několika fázích. V dnešní době je jich na střední oběžné dráze (Medium Earth Orbit, výška kolem 20 000km) přesně 31. Nejstarších 11 z nich bylo vypuštěno v letech 1990 až 1997. Jejich dráhy jsou umístěny tak, že v každém okamžiku kdekoli na zemi by mělo být možné chytit signál alespoň ze šesti družic. V ideálním případě z deseti**.** Za jeden den družice dvakrát oběhne okolo země. Jejich pozice je tak v daný čas na zemi každý den konstantní.
Poskytované GPS služby jsou rozděleny do několika sektorů. Satelit vysílá na několika frekvencích. Každému ze sektorů je přiřazena frekvence a jedna z nich byla roku 1997 zpřístupněna i veřejnosti.
K určení polohy na zemi je zapotřebí signálu z nejméně tří družic současně. S pomocí tří družic však ještě nelze určit nadmořskou výšku, k tomu jsou potřeba družice čtyři. Čím větší je počet družic, ze kterých má GPS přijímač signál, tím lépe, určení pozice je tak přesnější.
Detailní informace a vysvětlení jak přesně celý systém funguje lze nalézt na stránkách National Executive Committee for Space-Based Positioning, Navigation, and Timing (PNT) [8].

3.2. GPS přijímače

Přijímačů existuje velké množství a liší se hlavně jejich použitím, od kterého se také odvíjí jejich cena. Lze je rozdělit do několika skupin:

- Přijímače bez displeje. Tento typ vyžaduje současné použití s počítačem (nejčastěji přenosným). Komunikace ve většině případech probíhá pomocí technologie Bluetooth.
- Přijímače vybavené vlastním displejem. Není třeba použití dalších doprovodných zařízení. Mezi tyto typy patří například navigace používané v turistice, pro automobily, lodě a jinou dopravu. Často také bývají vybaveny externí anténou.
- Speciální přijímače. Do této skupiny spadají zařízení, která se používají pro zemědělství, GIS, armádou, atd. Většinou se jedná o jednoúčelové profesionální přístroje.

[8] Odkaz na specifikaci GPS. URL: http://pnt.gov/public/docs/2008/spsps2008.pdf

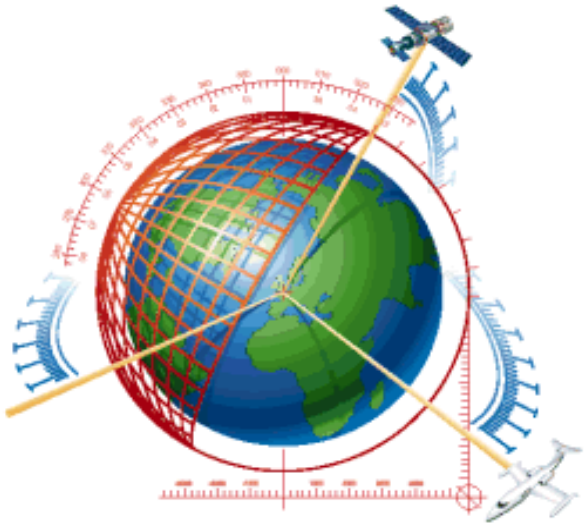
- Přijímač jako součást jiného celku. Zde patří především PDA a jiné kapesní počítače vybavené GPS přijímačem, které mají jinou primární funkci a přijímač je zde jen doplňkem.

3.3. Systémy a jejich promítání

Na to, aby bylo možno určovat pozici na zemi, je třeba zvolit nějaký geometrický útvar, na který se nám povrch Země promítne. Dále uvedu popis tří nejdůležitějších promítání vzhledem k území České republiky. Druhý z nich S-JTSK totiž jinde než u nás a na Slovensku použít nelze.

3.3.1. WGS (World Geodetic System)

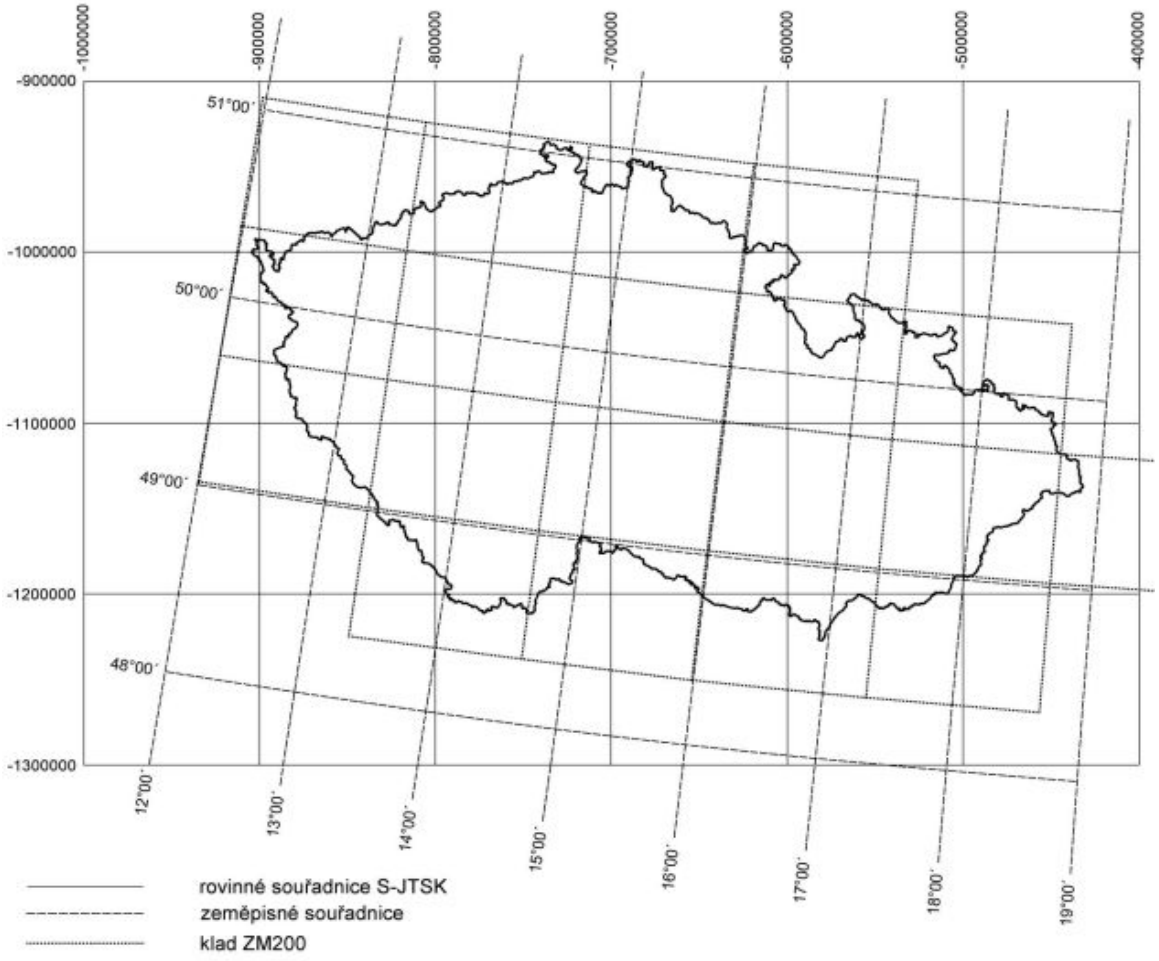
Standard souřadnicového systému vydaný USA roku 1984 a využívaný GPS. Mapa je promítaná na elipsoid. Poslední verze nese označení WGS84. Jedná se o pravotočivou kartézskou soustavu souřadnic, jejíž střed je v těžišti Země. Osa x směřuje k bodu, kde se protíná nultý poledník s rovníkem. Osa z směřuje k severnímu pólu a poslední y je kolmá na obě předchozí. Základními jednotkami jsou stupně (někdy udávané v dekadickém zápisu).
O těchto třech soustavách jsou zde uvedeny jen obecné informace. Pro detailní informace je možné využít dokumentů uvedených ve zdrojích této práce.



Obr. 3.3.1 – Země ve zobrazení WGS

3.3.2. S-JTSK

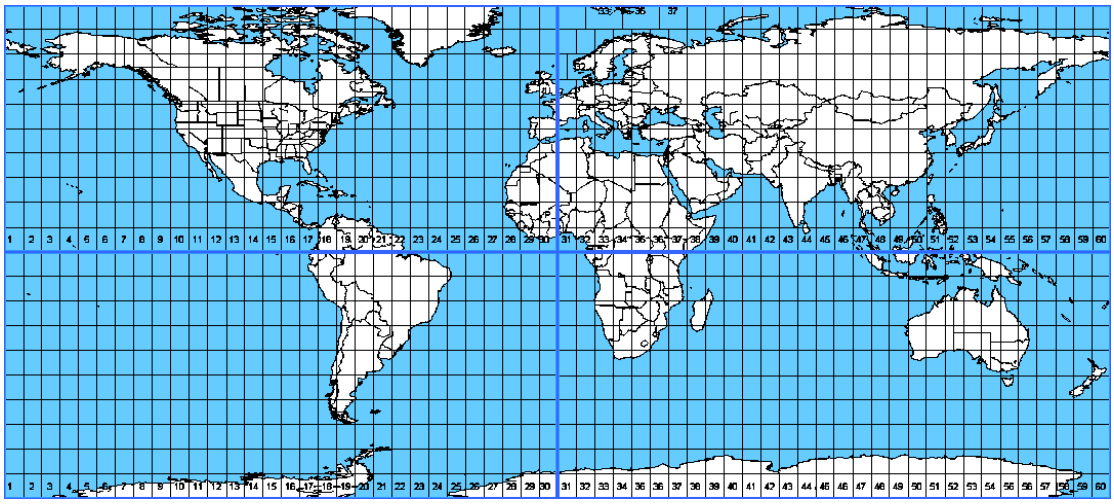
Jedná se o ryze česko – slovenské zobrazení. Vytvořeno Ing. J. Křovákem po první světové válce a někdy je také nazýváno Křovákovo zobrazení. Povrch Země je zobrazen na kužel v obecné poloze. S-JTSK bylo navrženo k použití výhradně na území České Republiky a Slovenska. Již při použití v přilehlých státech by docházelo k značné odchylce mezi vypočtenou a skutečnou pozicí. Osa x je obrazem poledníku zeměpisné šířky 42° 30’, osa y je na osu x kolmá a prochází obrazem vrcholu zobrazovacího kužele. Střed soustavy byl určen tak, aby odchylky na našem území byly co nejmenší. Základní jednotkou jsou metry. Souřadnice na našem území jsou tedy v kladných hodnotách a navíc se ani nedají zaměnit. Platí pro ně vztah x>y za každých podmínek.



Obr. 3.3.2 – Mapa ČR ve zobrazení S-JTSK

3.3.3. UTM (Universal Transverse Mercator)

Je historický nejstarší ze zde zmíněných a byl vyvinut armádou USA roku 1947. Země se promítá na příčné válcové zobrazení. Celé zobrazení je rozděleno na 60 oblastí, aby při



Obr. 3.3.3 – Mapa Země ve zobrazení UTM

určování polohy nedocházelo k velkým chybám. Určování polohy probíhá v jednotlivých zónách samostatně. Měří se v nich vzdálenost směrem k severnímu pólu a na východ. Vše je určováno v metrech.

Mezi všemi soustavami lze provádět transformace z jedné na druhou. Některé algoritmy jsou volně přístupné na internetu, za některé je třeba zaplatit.
Pro použití na 3D mapě jsem vybral systém S-JTSK, ze dvou důvodů. Celý systém je pravoúhlý a jeho základní jednotkou jsou metry. Nemusí tak docházet k přepočtům při zobrazování na mapě. Nevýhodou tohoto systému je, že ho lze použít pouze na území ČR a SR.

3.4. Zápis a zpracováni GPS dat

Existuje mnoho formátu, do kterých lze data zapsat. Liší se hlavně různorodostí dat, které lze uchovat. U jednoduchých formátů lze ukládat pozici jednotlivých bodů a z nich pak vytvářet polygony. Navíc můžeme mít dodatečné informace o jednotlivých bodech (čas uložení, popis místa, atd.). Nabízí se také možnost definovat vlastní formát, a tím i jeho obsah. Pro jednoduchost je ale lepší vybrat nějaký existující formát, který nejvíce vyhovuje daným potřebám svými dispozicemi, a také nalézt vhodný editor, ve kterém by bylo možné data upravovat. Volba formátu je důležitá. V případě, že ke zvolenému formátu je k dispozici také editor a konvertor do jiných formátů, dojde ke značnému ušetření práce a času. Vhodné je převést data do některého z formátů vektorové grafiky. Tyto formáty jsou ve většině případů podporovány v aplikacích pro tvorbu 3D grafiky. Jejich importem si lze celý proces značně zjednodušit. Uvádím zde jednoho zástupce formátu vektorové i rastrové grafiky.

SVG (Scalable Vector Graphics)

Jedná se o specifikaci vydanou W3C. Definuje funkce a syntaxi k tvorbě vektorové i rastrové grafiky pomocí XML. Tento formát je obsáhlý a na stránkách W3C[9] si lze prohlídnout jeho úplnou specifikaci. Pro potřeby této práce bude využito pouze generování ploch a křivek. V příloze A je uvedena část SVG souboru s krátkým popisem. Tento soubor jsem vytvořil na základě GPS dat ze zdroje OpenStreetMap[10].

Vhodným editorem k tomuto formátu může být například aplikace Inkscape. Jedná se o volně šířitelný software. Jeho hlavním zaměřením je právě typ souboru SVG, podporuje však i další formáty vektorové i rastrové grafiky.

Této aplikace jsem využil pouze k prohlížení SVG souborů.

3.5. Zdroje GPS dat

Získání dat může proběhnout několika způsoby. Některé z nich jsou časově náročné jiné méně. Data lze získat již hotová (zpracovaná někým jiným) nebo je nutné provést vlastní sběr dat. Proces sbírání dat může být i docela komplikovaný. Záleží na několika faktorech. Požadovaná přesnost dat, objem dat (dodatečné informace o lokalitě) a velikost oblasti, pro kterou se sběr provádí. Dále je uveden krátký popis tří možností, jak data získat.

3.5.1. Sběr dat v terénu

K provedení sběru dat v terénu je zapotřebí GPS přijímač a aplikace, která bude informace zaznamenávat. Nabízejí se zde jak aplikace komerční, tak freeware. Důležité je uvědomit si, jaký bude objem dat, které potřebujeme zaznamenat. Jedná-li se o dvě budovy, pak není nutné využití žádného programu a lze si vystačit s poznámkovým blokem a GPS přijímačem. Pro sběr dat velkého obsahu je potřeba aplikace a vytvoření struktury sbíraných dat. Důležitou věcí je otázka přesnosti. K dosažení co nejlepších výsledků je třeba vybrat dobře denní dobu, abychom v daném čase na daném místě měli ve výhledu co největší počet družic. Vyhovující je počet sedmi až osmi družic ve výhledu. Dalším faktorem je počasí. Měření by se pokud možno mělo provádět za dobrého počasí (obloha bez mraků). Správný počet družic s dobrým rozmístěním po obloze a dobré počasí, jsou základem úspěšného měření. Dále je třeba se soustředit na samotné měření. Existuje několik metod, jak měření provádět. Metoda průměrování znamená, že se zůstává stát na jednom místě delší čas a zaznamenává se několik vzorků souřadnic. Výsledná poloha je pak stanovena z průměru naměřených vzorků. Jednou z dalších možností je využití referenčních GPS stanic, které jsou rozmístěny na území ČR. Vždy je nutné vybrat stanici, která má nejmenší vzdálenost od měřeného území (maximální vzdálenost asi 50km). Stanice mají pevně stanovenou GPS pozici. V průběhu dne stanice pořizují reálné naměřené hodnoty, které po odečtení od jejich stanovené pozice dají odchylku. Tyto odchylky je možné ze stanice získat a pomocí nich opravit vlastní naměřená data. Tuto opravu lze provádět přímo při samotném měření, nebo později. Přímý způsob vyžaduje připojení k internetu v terénu.

[9] Specifikace SVG formátu, URL www.w3.org/TR/SVG11/index.html

[10] Postup při zpracování dat ze serveru viz. kapitola 7.1.1 Data ze serveru OpenStreetMap

3.5.2. Hotová data

Hotovými daty se rozumí data, která již zpracoval někdo jiný. Je dobré prohledat internet, protože je možné, že už někdo lokalitu, o kterou se zajímáte, zpracoval. Otázkou zůstává, jak přesné a detailní informace jsou. Nejjednodušším způsobem je ověřit si malou část dat vlastním měřením. Velmi dobrým zdrojem informací je OpenStreetMap. Všechna data jsou volně ke stažení a nahrávají je zde samotní uživatelé. Mapy lze prohlížet online nebo je stáhnout do PC. Je zde i několik exportů do různých datových či obrázkových formátů.

3.5.3. Data z mapy

Data lze získávat také z foto map nebo satelitních snímků. Zde je potřeba zvolit několik referenčních bodů, u kterých je známa pozice. Tyto body poslouží ke korekci mapy a také jako kontrola, je-li mapa ve správném měřítku a projekci. Mapu lze umístit na pozadí v libovolném grafickém editoru a překreslit. Důležitá je projekce mapy. Snímky, které využívají WGS systému nelze použít pro systém UTM. Například GoogleMaps využívají UTM systému a GoogleEarth zase WGS systému. Tuto skutečnost si lze snadno ověřit. Při pohledu na celou mapu (maximálně zmenšenou) je důležitý její tvar. Obdélník značí použití systému UTM (plocha válce v rovině) a má-li tvar kruhu jedná se o systém WGS.

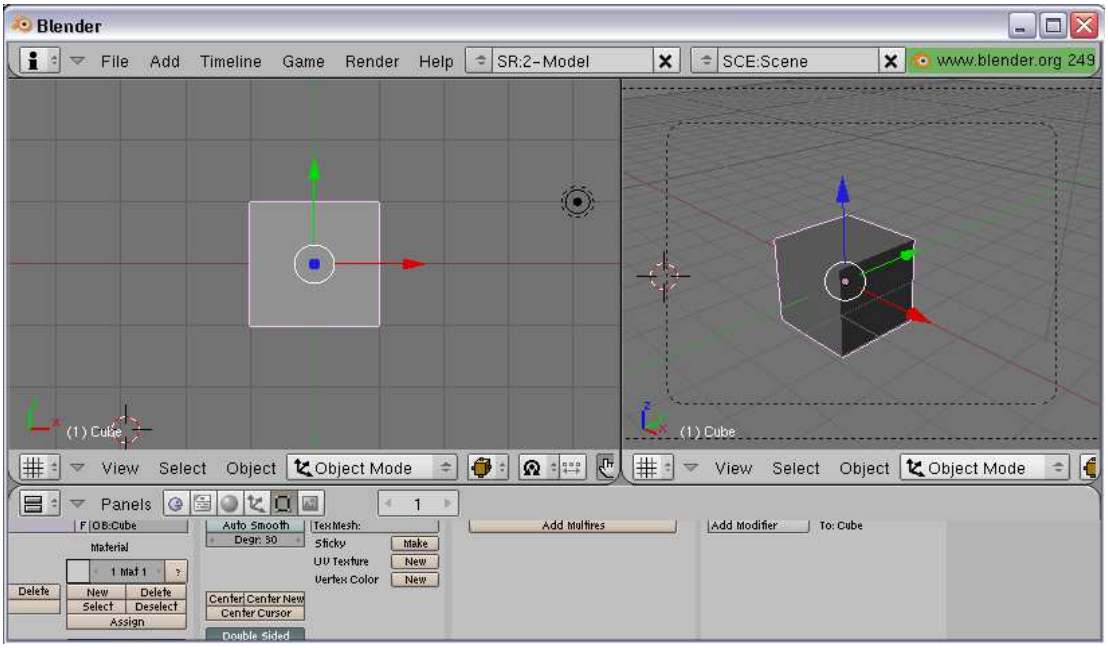
4. Aplikace pro tvorbu 3D scény

Tato kapitola pojednává o aplikacích, ve kterých lze vytvořit 3D model. Většina aplikací poskytuje širokou paletu funkcí a jejich zvládnutí zabere velké množství času. Je možné vytvářet a editovat 3D objekty, různě je animovat, měnit jejich fyzikální vlastnosti, atd. Navrhovaná aplikace bude obsahovat statický 3D model, který bude možné importovat a zobrazit pod DirectX. Nebude využívat žádných animací, ani modelovaní pomocí částic, jen základní práci s polygony a následné mapování textur. Model musí mít co nejmenší počet polygonů při zachování jeho vizuální věrohodnosti, aby nenastaly komplikace při spouštění a zobrazování na mobilním zařízení (malá velikost paměti a pomalý procesor). Důležitým faktorem je, zda-li bude aplikace disponovat exportem do X file[11], protože pomocí tohoto formátu lze dále model převádět až do výsledného formátu MD3DM[11].
Neuvádím zde přehled všech dostupných aplikací a jejich vlastností, ale jen několik významných zástupců. Mezi nejznámější patří 3ds Max, Blender, Maya, SketchUp a TrueSpace. Aplikace se dají rozdělit na freeware (volně šiřitelné) a shareware (komerční). Do komerční skupiny spadá 3ds Max a Maya. Všechny tyto aplikace jsou schopny posloužit účelům této práce. Jedná se však o profesionální aplikace, jejichž cena se pohybuje v desetitisících korun. Pozornost je proto věnována spíše volně šiřitelným aplikacím, u kterých je hlavním problémem chybějící export do X file. Tyto dvě aplikace patří mezi volně šiřitelnými k těm nejpoužívanějším.

4.1. Blender

Patří na špici mezi volně šiřitelnými aplikacemi pro 3D modelování. Byl vytvořen dánskou společností NeoGeo jako komerční software ke konci devadesátých let. Zajímavostí na Blendru je, že roku 2002 byl převeden na GNU (General Public License). Společnosti NeoGeo bylo jednorázově vyplaceno 100 000 euro a Blender je nyní volně šiřitelný. Peníze byly vybrány ve sbírce, na kterou mohl přispět úplně každý. Dohled nad vývojem dodržuje Blender Foundation.
Blender byl původně navržen k profesionálnímu použití. Jeho rozhraní je na první pohled komplikované (obr. 4.1). Vše je však navrženo tak, aby jednotlivé úkony zabraly co nejmenší čas. Základem je mít jednu ruku na myši, zatím co druhá ruka obsluhuje klávesnici. Veškeré funkce jsou dostupné pomocí klávesových zkratek. Zvláštností Blendru je, že nepoužívá takzvaných PopUp oken. Celá uživatelská plocha je rozdělena na několik částí, které si může uživatel libovolně upravovat (přemístit, rozdělit, odebrat). Tyto části se nemohou za žádných okolností překrývat. Například při ukládání souboru, kdy drtivá většina programů použije PopUp okno, Blender zobrazí část rozhraní pro uložení v některé z jeho částí. V Blendru je mimo základní modelovací nástroje možnost použití i nástrojů pro tvorbu animací a her. Importovat a exportovat lze velké množství formátů. Požadovaný X file je jedním z nich. Jedná se o velice mocný nástroj, který nabízí funkce drahých profesionálních modelovacích nástrojů, ke kterým se běžný uživatel dostane jen zřídka.

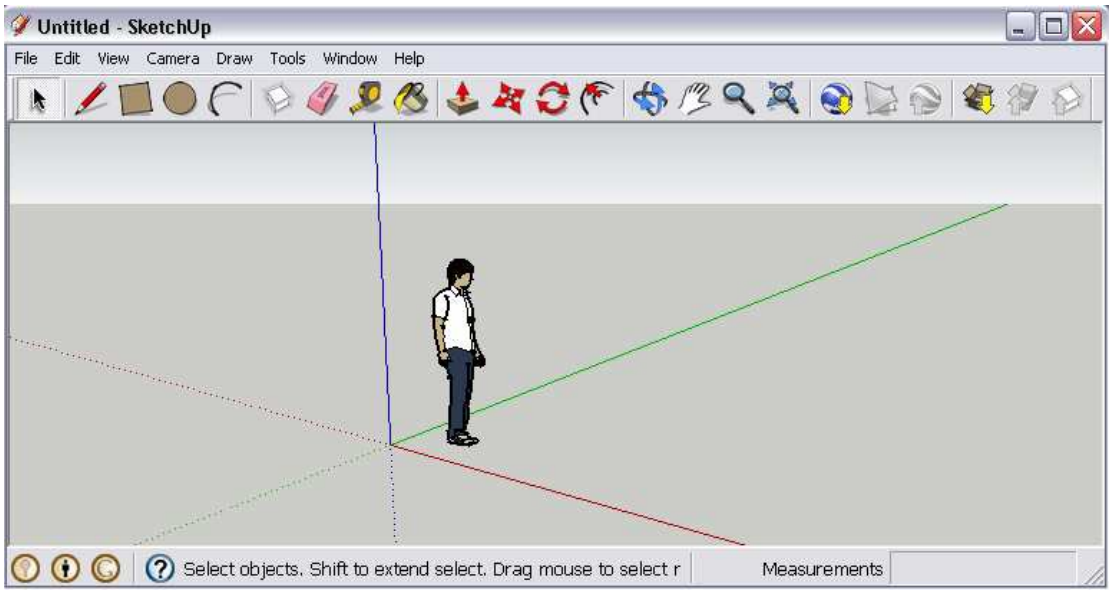
[11] Popis formátů X file a MD3DM viz. kapitola 6.2 Direct3D Mobile – MeshLoader (X File)



Obr. 4.1 – Uživatelské rozhraní aplikace Blender

4.2. SketchUp

Je projektem společnosti Google a jedná se o relativně novou aplikaci. Po prvé se objevila roku 2000. Vydává se ve dvou verzích. Na základní verzi se vztahuje freeware licence. Komerční verze nese označení Pro. Hlavní filozofií SketchUp je jednoduchost. Přístup k tvorbě 3D objektů je naprosto jiný ve srovnání se všemi ostatními aplikacemi. Klávesové zkratky se moc nepoužívají a všechny operace se provádějí pomocí tlačítek. Jedná se o velice zajímavé řešení a je tou nejlepší volbou pro uživatele, kteří nechtějí strávit hodiny



Obr. 4.2 – Uživatelské rozhraní SketchUp

seznamováním se s uživatelským rozhraním aplikace. Obr. 4.2 zobrazuje rozhraní tohoto programu hned po spuštění aplikace. Na první pohled si lze povšimnout jednoduchosti celého rozhraní. Požadovaný export do X file však u nekomerční verze chybí a je implementován pouze u verze komerční.

K vytvoření modelu bude použito aplikace Blender[12]. Při použití SketchUp, by bylo zapotřebí jeho komerční verze z důvodu chybějícího exportu do X file. Dalším důvodem je, že Blender nabízí profesionální prostředí. K oběma programům lze nalézt množství tutoriálů na internetu a v knihách.

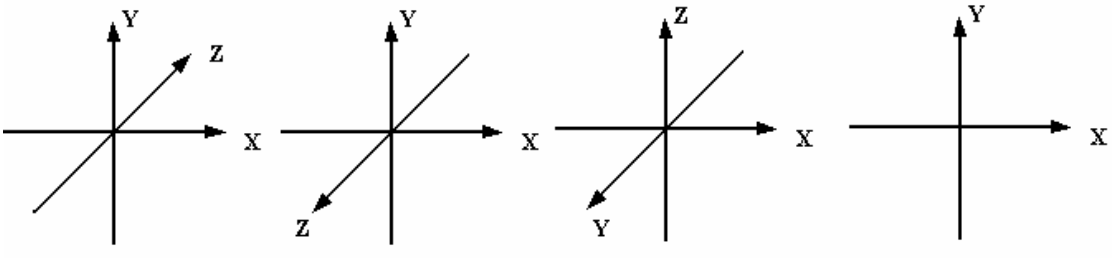
[12] Postup vytváření 3D modelu viz. kapitola 7.2 Tvorba modelu.

5. Základy 3D grafiky

K vytvoření 3D modelu a naprogramování aplikace, která ho bude zobrazovat je nutnost znalosti základů 3D grafiky. Když jsem začal psát tuto diplomovou prácí, tak jsem měl velmi malé zkušenosti s 3D grafikou. Rozhodl jsem se proto věnovat jednu kapitolu vysvětlením této problematiky. Informace jsou záměrně uvedeny v samostatné kapitole, protože jsou obecně platné a nevztahují se tak přímo k některé z knihoven pro tvorbu 3D aplikací.

5.1. Koordinační systémy 2D a 3D

K zobrazování objektů v prostoru se využívá koordinačních systémů. Několik z nich je předvedeno na obr. 5.1. První zleva se nazývá pravoruký a je využíván například knihovnou OpenGL. Druhý v pořadí je systém levoruký. Tohoto systému využívá právě Direct3D. Jediným rozdílem mezi těmito systémy je orientace osy z, která je opačná. V další části obrázku je znázorněn systém využívaný zejména mezi programy pro tvorbu 3D modelů (Blender, SketchUp a další). Poslední ze systému slouží k vytváření 2D grafiky a je využíván například knihovnou DirectDraw.



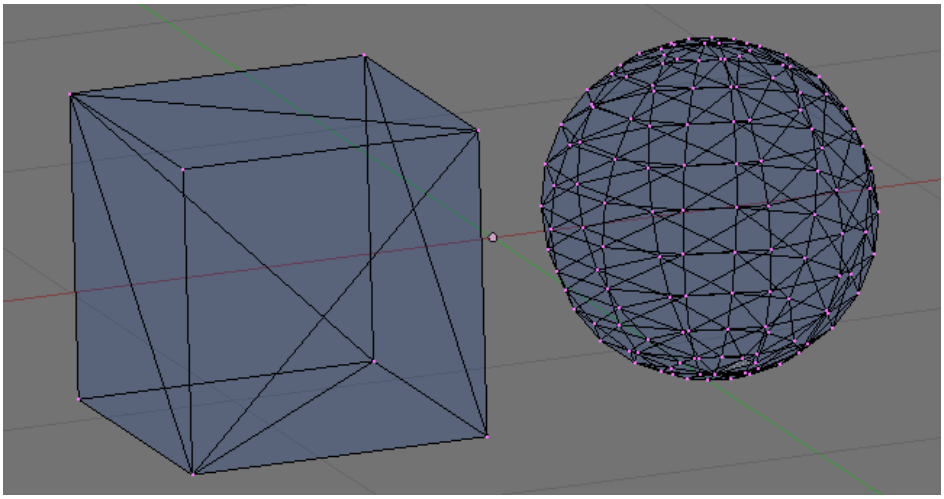
Obr. 5.1 – Koordinační systémy

5.2. 3D objekty

Základním stavebním kamenem každého 3D objektu je bod, neboli vertex. Ten je ve 2D prostoru určen souřadnicemi x a y a ve 3D prostoru je zde ještě třetí souřadnice z. Pomocí dvou bodů lze vytvořit úsečku a pomocí tří bodů trojúhelník, jež je základním geometrickým útvarem pro tvorbu 3D modelů. Trojúhelník je také často nahrazován slovem polygon. Veškeré modely a struktury jsou vytvářeny pomocí tohoto základního útvaru. Obr. 5.2 zachycuje takto vytvořenou krychli a kouli.
Z důvodu snadné tvorby a manipulace s těmito objekty se většinou k jejich vytvoření využívá několika struktur popsaných níže. Každá z těchto struktur v podstatě uchovává pole bodů a jejich souřadnice. V závislosti na typu struktury nám vznikají rozdílné objekty.

Popis jednotlivých struktur:

- List bodu (Point List) – Dojde k vytvoření šesti samostatných bodů.
- List úseček (Line List) – Tři úsečky tvořeny po sobě jdoucími body.
- Pás úseček (Line Strip) – Pět navzájem spojených úseček.
- List trojúhelníků (Triangle List) – Dva, na sobě nezávislé trojúhelníky tvořeny trojicí po sobě jsoucích bodů.
- Pás trojúhelníků (Triangle Strip) – Čtyři spojené trojúhelníky. První je určen trojicí prvních bodu, další pak dvojicí posledních bodu z předchozího trojúhelníku a jedním novým bodem. Takto se postupuje až k poslednímu bodu ve struktuře.
- Vějíř trojúhelníků (Triangle Fan) – Velice podobný pásu trojúhelníků s jedním rozdílem. Po vytvoření prvního se další trojúhelníky tvoří pomocí prvního a posledního bodu z předchozího trojúhelníku, doplněných jedním novým bodem.



Obr. 5.2 – Objekty vytvořené pomocí polygonů

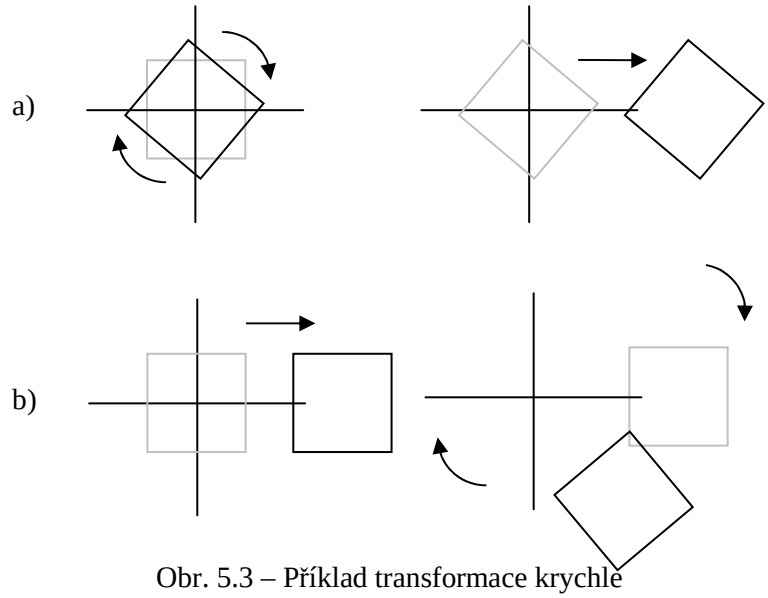
Záleží na pořadí zadávání bodů, které by měly být zadány po směru hodinových ručiček (clockwise), aby byl výsledný trojúhelník viditelný. Protože každý z nich má přední a zadní stranu. Pro urychlení výpočtu a zobrazení scény jsou trojúhelníky, které jsou k pozorovateli otočeny zády zanedbávány.
Každý vertex může mít přiřazenu svoji normálu, které se využívají k výpočtu osvětlení scény[13].

5.3. Transformace

Veškeré transformace se ve 3D prostředí provádějí za pomocí matic, jsou tedy velice důležité. Patří zde rotace, posuny, zvětšení a zmenšení podle všech tří os, ať už jednotlivě, nebo

[13] Popis osvětlení viz. kapitola 5.5 Osvětlení

dohromady. Transformace na maticích jsou v knihovnách pro tvorbu 3D aplikací proveditelné pomocí metod, které danou matici třeba s rotací podle osy x vytvoří. V případě, že se na objektu provádí několik operací (rotaci podle osy x, zmenšení a posunutí po ose z) najednou, je z několika matic vytvořena matice jediná, která plní stejnou funkci. Tuto matici lze získat vynásobením dílčích matic. Zde však nutné dbát na pořadí, ve kterém se dílčí operace mají provádět. Obr. 5.3 znázorňuje dvě situace, na kterých byly provedeny tytéž operace, ale bylo prohozeno jejich pořadí (a – rotace a pak posun, b – posun a pak rotace).



Obr. 5.3 – Příklad transformace krychle

5.4. Kamera

Pohled na scénu je zajišťován kamerou. K jejímu nastavení patří určení její polohy, směru, kterým se kamera dívá, její úhel záběru, dosah (vzdálenost, za kterou budou objekty ignorovány) a úhel natočení.

5.5. Osvětlení

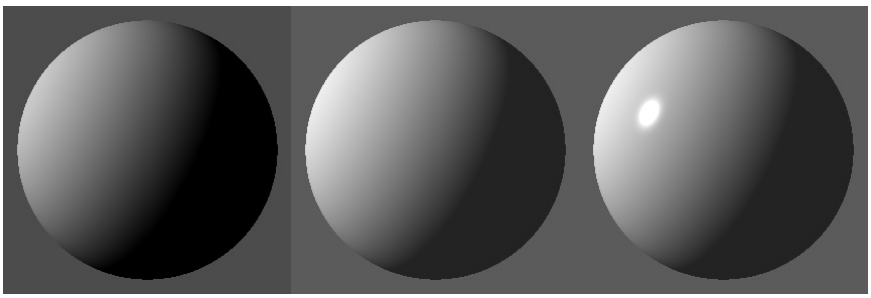
Světlo v reálném světě je tak komplikované, že by jej nebylo možno v reálném čase zobrazovat. Jednotlivé složky světla se od všech předmětů okolo odráží nebo jsou pohlcovány. Dojde tak k nepočitatelnému počtu odrazů, které v počítačové grafice není možné simulovat. Proto je světlo rozděleno na soustavu tří složek, pomocí kterých se lze k vlastnostem reálného světla alespoň přiblížit.

Složky světla a jejich krátký popis:

- **Ambient** – Složka, která rovnoměrně osvětluje celou scénu. Nemá žádný zdroj ani směr a všechny plochy objektů jsou osvětleny naprosto stejně. Na denním světle si lze povšimnout, že i objekty ve stínu jsou viditelné. Světlo se k nim dostává odrazem. Tuto složku reálného světla lze nahradit ambient osvětlením.

- **Diffuse** – Přímo osvětluje scénu, nemá žádný zdroj, ale má směr. Tímto světlem lze simulovat například přímý sluneční svit.
- **Specular** - Poslední složkou světla, která má zdroj i směr. Používá se k simulaci odlesku na objektech.

Příklad jak jednotlivé složky působí na scénu zobrazuje sekvence na obr. 5.5 Složky byly postupně přidávány v tomto pořadí: diffuse, ambient, specular. V druhé části obrázku si lze povšimnout lehkého osvětlení spodní části koule zajištěného složkou ambient a v poslední části zase nejsvětlejší části vytvořené složkou specular.



Obr. 5.5 – Vliv jednotlivých složek světla na scénu

Jednotlivé knihovny pro vývoj 3D aplikací si definují vlastní sadu světel[14], pomocí které lze scénu osvětlovat. U těchto světel se mimo jejich parametrů (směr, intenzita, atd.) nastavují také barvy pro jednotlivé složky (diffuse, ambient, specular). Pomocí nich se určuje, jakou barvu budou jednotlivé složky světla vyzařovat.

Při vykreslování objektů lze každému z nich přiřadit materiál, který definuje barvy pro jednotlivé složky světla (ambient, diffuse a specular), které bude objekt odrážet.

Normály vektorů slouží k určení normály pro jednotlivé polygony, které jsou na plochu polygonu kolmé. Pomocí nich se určuje, pod jakým úhlem světlo dopadá na jednotlivé polygony. V závislosti na velikosti úhlu dopadu se osvětluje daný polygon (s rostoucím úhlem klesá intenzita světla).

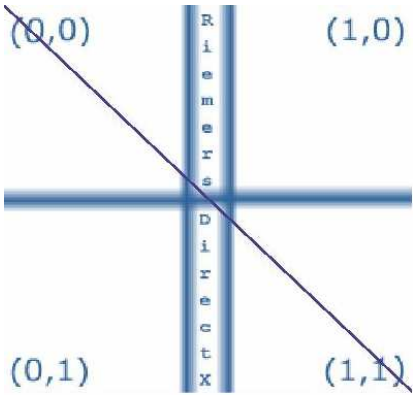
5.6. Barvy

K definování barev se používá výhradně modelu RGB a všechny barvy jsou tak tvořeny kombinacím červené, zelené a modré barvy. K dosáhnutí toho, aby se objekt jevil průhledný, se používá alfa barvy. Ta říká na kolik je daný objekt průhledný (255 vůbec až po 0, kdy je objekt úplně průhledný). Další možností jak dosáhnout efektu průhlednosti je míchaní barev. Smícháním barvy objektu s jeho pozadím lze docílit průhlednosti. Tato operace se nazývá color bleading.

[14] Popis světel u DirectX Mobile viz. kapitola 6.2 Direct3D Mobile - Lights

5.7. Textury

Textury jsou mnohem hojněji užívané oproti obarvení modelů barvami. Objekty obarvené jednou barvou vypadají velice plastiky a nereálně. Textury naopak dodávají objektům na reálnosti. Zpravidla se ukládají v samostatném souboru a záleží na jednotlivých modelovacích prostředích, jaké formáty obrázku jsou podporovány. Nejdůležitější na texturách je jejich mapování. Textura je rozdělena pomocí souřadnic, kde maximální x i y souřadnice můžou nabývat hodnot z rozsahu 0 až 1. Obr. 5.7 ukazuje příklad textury a jeho mapování na čtverec složený ze dvou trojúhelníků.



Obr. 5.7 – Textura na čtverci

5.8. Mesh

Jedná se o strukturu, která uchovává veškeré informace o 3D objektu (body, normály bodů, polygony, odkaz na texturu a mapování textur). Jedná se o jakýsi systematický celek, který usnadňuje práci a manipulaci s 3D objekty. Existuje velké množství formátů, do kterých se dá mesh uložit. Informace jsou v mesh rozděleny na bloky. Například blok bodů zahrnuje všechny body. Dalším blokem může být blok hran, který pro jejich vytvoření využije blok bodů. Z hran se dále dají poskládat strany a to bude uloženo v dalším bloku. Obsahuje také odkaz na texturu a způsob, jakým bude textura aplikována (mapována) na jednotlivé polygony. Při tvorbě modelu a aplikace jsem využil formátů X file a MD3DM[15].

5.9. Back Buffers

Back buffer je nedílnou součástí každé 3D aplikace, která zajišťuje plynulý pohyb celé scény (animace) a zabraňuje blikání obrazovky. Celý systém je založen na jednoduchém principu.
Je použit přední buffer, jehož obsah je vždy zobrazován na displeji. Nová scéna se nakreslí do zadního bufferu, který se po dokončení vykreslování zobrazí na přední buffer. Tento systém nikdy neumožňuje přístup k zápisu do předního bufferu. Vždy se zapisuje jen do toho

[15] Popis formátů X file a MD3DM viz. kapitola 6.2 Direct3D Mobile - MeshLoader

zadního. Při použití jednoho zadního bufferu se metoda nazývá Double Buffering. U knihoven DirectX a OpenGL je Double Buffering implicitně zapnut. Je zde však i možnost použití více než jednoho zadního bufferu, jež značně pomáhá ke zlepšení plynulosti aplikace. Zejména u her a u aplikací s rychlým pohybem.

5.10. Depth Buffer

Někdy je také nazýván Z buffer. Jeho hlavním úkolem je sledovat jakou mají jednotlivé objekty vzdálenost od kamery. Při renderování by bez tohoto bufferu docházelo k tomu, že by se objekty vykreslovaly v pořadí, ve kterém je zadáme. Mohlo by se stát, že objekt, který je ve větší vzdálenosti od kamery nám překreslí objekt s menší vzdálenosti. U 2D aplikací se toho bufferu většinou nevyužívá.

5.11. Projekce

Nejpoužívanější jsou dva druhy projekce a to ortogonální a perspektivní. Ortogonální projekce zanedbává vzdálenost od kamery v tom ohledu, že objekty stejné velikosti s jinou vzdálenosti od kamery budou mít stejnou velikost i na vykreslené scéně. Perspektivní pak reprezentuje klasický pohled, jako přes čočky kamery nebo fotoaparátu. Objekty se s rostoucí vzdáleností od kamery zmenšují.

6. D3D Mobile a .NET CF

6.1. .NET Compact Framework

Tato část diplomové práce popisuje součástí knihovny .NET Compact Framework. Zahrnul jsem zde pouze třídy, které jsem využil při tvorbě aplikace. Jejich popis je pouze obecný a zachycuje hlavně využití jednotlivých tříd.

System.Windows.Forms.Form

Grafické rozhraní, které slouží jako hlavní zobrazovací jednotka, jehož výstupem je okno. K jeho konstrukci včetně komponent lze přistoupit dvěma způsoby. Je zde možnost využít návrhového pomocníka, který je součástí Visual Studia, nebo všechen kód napsat. Práce s návrhovým pomocníkem je velice jednoduchá a rychlá. Do okna lze vkládat různé komponenty ať už přímo, nebo lze komponenty umisťovat do kontejnerů (*Panel*) a ty pak umístit do okna. Toto řešení se vyplatí zejména, když zde bude nějaký panel tlačítek, která by měla zůstat stále pohromadě. Takto pak stačí přemisťovat kontejner a s ním se bude pohybovat i celý jeho obsah.
Třída *Form* disponuje událostmi, které upozorňují na to, že došlo k přechodu do nového stavu: aktivní (*Activated*), uzavřen (*Closed*), má-li fokus nebo ne (*Focused*), atd. Událost, kdy *Form* přejde do neaktivního stavu se dá využít k pozastavení aplikace. Uživatel pravděpodobně pracuje s jinou aplikací a je rozumné uvolnit část paměti a zbytečně nezatěžovat zařízení tím, co stejně není vidět. Dále se tímto řešením bude šetřit baterie. V situaci, kdy uživatel zanechá mobilní zařízení ležet bez povšimnutí několik desítek hodin se spuštěnou aplikací, tak nedojde k vybití baterie, které by jinak hrozilo. Další důležité události jsou zděděny ze třídy *System.Windows.Forms.Control*, pohyb, stisk a uvolnění tlačítka myši. Cílové mobilní zařízení, pro které bude aplikace vytvářena, by mělo obsahovat dotykový displej. Na každý dotyk displeje prstem reaguje právě událost myši. Při přiložení prstu na displej se nejdříve vyvolá stisk tlačítka myši (*MouseDown*)(pokaždé se jedná o levé tlačítko). Tažením prstu po displeji se vyvolá událost pro pohyb myši (*MouseMove*) a po odtažení prostu od displeje nastane poslední událost uvolnění tlačítka myši (*MouseUp*). S těmito třemi událostmi si lze vystačit k obsluze displeje jako vstupního zařízení, které bude přebírat příkazy od uživatele.

Komponenty System.Windows.Forms

Všechny komponenty dědí ze třídy *System.Windows.Forms.Control* a dále se dělí na několik skupin (tlačítka, seznamy, obrázky atd.). Proto je také většina nastavitelných parametrů totožná (velikost, pozice, viditelnost, název, barva pozadí atd.). U všech lze také využít události vstupu z klávesnice nebo myši. Dále následuje popis komponent, které jsem využil v této diplomové práci.

Button – Klasická funkce tlačítka. Tuto třídu zde uvádím jen proto, že se v knihovně Compact Framework pozapomnělo na funkci přiřazení obrázku na pozadí tlačítka. U desktopové verze tato funkce je a nevidím důvod proč nebyla implementována i zde. Design aplikace tak může hodně získat. Tlačítka znázorňující jen text na nějakém jednobarevném podkladu vypadají nevýrazně a jednoduše.
UserControl – Slouží programátorovi k vytvoření vlastní komponenty. K použití tohoto prvku může vést několik důvodu. Potřeba zcela jiné funkce komponenty než je k dispozici nebo tvorba komponenty s vlastním originálním vzhledem.
ListBox – Zobrazení seznamu lze provést právě pomocí této komponenty. Dá se zde zobrazit jakýkoli seznam objektů, který pak uživatel může procházet a zvolit libovolnou položku seznamu. Komponentě se přiřazuje seznam objektů (např. *List* nebo *ArrayList*) a atribut, podle kterého se bude obsah prezentovat uživateli. Pro určení, která položka byla vybrána jsou zde metody vracející přímo vybraný objekt (*SelectedItem*) nebo jen jeho index (*SelectedIndex*)(polohu v seznamu).
WebBrowser – Komponenta zobrazující jakýkoli HTML obsah, který lze definovat přímo vložením řetězce typu *String* nebo pomocí URL. Je vhodná zejména pro zobrazování většího obsahu textu, který obsahuje různá písma a zároveň také obrázky. Hlavní výhodou této komponenty je jednoduchost jejího použití. V situaci, kdy je potřeba zobrazit kombinaci textu s obrázky by jinak bylo nutné využít komponent *Label* a *PictureBox*. Jejich použití pro větší objem zobrazovaných dat by byl špatně organizovatelný a zdlouhavý.
Poslední zmíněnou skupinou jsou objekty pro tvorbu menu. Jedná se o klasické menu, tak jak je známo u většiny mobilních zařízení, které se nachází ve spodní části displeje.
Komponenty pro tvorbu menu:
MainMenu – Třída, která slouží jako základ menu. Nově vytvořené menu je třeba přiřadit nějaké třídě *Form*. Po vytvoření nic neobsahuje a vytvoří na spodní části obrazovky pruh (barva tohoto pruhu je závislá na nastavení ve Windows a v programu se nedá změnit).
MenuItem – Zastupuje položku menu. Každá položka může být přiřazena, buď přímo do základu menu (*MainMenu*) nebo nějaké jiné položce (*MenuItem*). Tímto způsobem se dají tvořit kaskádová menu. U položek je možnost nastavit jen název, který se bude zobrazovat uživateli. Nic jiného, co se vzhledu týče zde k nastavování není.

System.Drawing

Třída zapouzdřující vše zabývající se jednoduchou vektorovou grafikou (2D). Z této třídy jsem využil práce s obrázky, jednoduché datové struktury (bod – *Point*, obdélník - *Rectangle*, velikost - *Size*), a také práce s textem a barvami.

System.Xml

Výborný pomocník při práci s XML soubory, který umožňuje jejich procházení, tvorbu a editaci. Součástí této třídy je také možnost serializace objektů pomocí XML. .Net Compact Framework nepodporuje automatickou serializaci objektů, tak jak je tomu u desktopové

verze. Jako náhradu lze použít Compact Formatter[16] pod licencí LGPL, který plně nahrazuje chybějící část .Net Compact Framework.
Použití serializace, která se provede do XML souboru má i své výhody. Do souborů se dá snadno nahlížet a také jej editovat. V případě importu informací do aplikace je zde možnost nabídnout uživateli strukturu XML souboru, pomocí kterého lze import provést. Mobilní aplikace tak nemusí být doprovázená žádnou aplikací desktopovou, která by data od uživatele převáděla.

Microsoft.WindowsMobile.Samples.Location

Tato třída není oficiální součástí .Net Compact Framework, ale součástí vzorových příkladů. U Windows Mobile 5 SDK se v kódu nacházelo značné množství chyb. S příchodem novější verze 6 se Microsoft snažil všechny chyby opravit. Při testování třídy obsažené ve verzi 6 jsem narazil na chyby pouze ve formuláři demonstračního příkladu, který této třídy využíval. Vzhledem k tomu, že po otestování vše fungovalo v naprostém pořádku, upřednostním použití této již hotové třídy. Další možností by bylo napsat vlastní ovladač.
Třída zapouzdřuje všechny operace pro využití GPS přijímače. Jsou zde metody na uvedení GPS přijímače do chodu a jeho vypnutí. Třída vyvolává dvě události. První z nich je vyvolána, když zařízení přechází do jiného stavu (vypnuto, zapnuto, vypíná, zapíná). Druhá pak při jakékoli změně údajů obdržených z družic. Informace, které třída poskytuje, jsou více než dostačující. Jsou zde všechny základní údaje, jako je zeměpisná šířka, délka, nadmořská výška, počet satelitů ve výhledu a čas. Dále je zde rychlost, ukazatele kvality signálu co do počtu satelitů a jejich rozmístění po obloze a detail každého ze satelitů (jeho poloha a síla signálu).
Výčet informací, které lze pomocí této třídy ze satelitu získat:

- **Latitude** – Zeměpisná šířka udává se ve stupních nebo v decimálním zápisu.
- **Longtitude** - Zeměpisná délka udává se ve stupních nebo v decimálním zápisu.
- **Sea Level Altitude** – Nadmořská výška v metrech.
- **Ellipsoid Altitude** – Výška nad elipsoidem v metrech.
- **Heading** – Úhel kterým směrujeme v radiánech. Sever je roven 0.
- **Satellite Count** – Počet satelitů od kterých je příjem signálu.
- **Satellite In View Count** – Počet satelitů ve výhledu.
- **Speed** – Rychlost v uzlech.
- **Time** – Určení času.
- **Horizontal Dilution Of Precision**, **Vertical Dilution Of Precision**, **Position Dilution Of Precision** – Pomocí těchto tří hodnot se určuje kvalita přijímaného signálu.

Lokalizace

.NET Framework řeší změnu jazyka pomocí třídy System.Resources.ResourceManager, pomocí které lze načítat externí soubory (assembly) obsahující texty v daném jazyce. V praxi

[16] Compact Formatter, URL: http://www.freewebs.com/compactFormatter/index.html

to pak vypadá tak, že existuje jeden soubor pro každý *Form* nebo jen jeden, ve kterém jsou uloženy všechny texty v jednom jazyce. Pro každý jazyk se vytváří nový soubor. Názvy souborů jsou přesně definovány a například pro Americkou angličtinu se musí soubor jmenovat .en-US.resx, pro češtinu pak .cs-CZ.resx. V tvorbě těchto souborů vypomáhá vývojové prostředí Visual Studio .NET. V nastavení jednotlivých *Form* komponent stačí přepnout jazyk a přepsat všechny texty do požadovaného jazyka. Soubory s informacemi jsou automaticky vytvořeny.
Změna jazyka probíhá u většiny programů automaticky na jazyk, který využívá Windows OS. Nabízí se však i možnost jazyk kdykoli změnit. U .NET Compact Framework tato možnost není a jedinou možností, jak docílit, aby aplikace změnila jazyk, je změnit jazyk celého OS. U desktopové verze Windows by to znamenalo přeinstalování celého systému. Mobilní zařízení s Windows Mobile je nutné resetovat a u většiny je volba jazyků omezená. V případě vývoje aplikace, kterou budou uživatelé používat na svých zařízeních nedojde k žádnému problému. Předpokladem bude, že každý uživatel dobře zná jazyk, ve kterém operuje jeho mobilní zařízení. Problém nastává v případě, kdy si budou uživatelé vypůjčovat aplikaci i se zařízením. Jediným řešením toho problému je nevyužívat lokalizace, kterou nabízí .NET Compact Framework a vytvořit vlastní strukturu pro ukládaní jazyků. Na internetu jsem našel pár balíčků od různých programátorů, kterými lze změnu jazyka provádět. Jako příklad zde uvádím jeden dobrý článek s názvem User Interface Localization with the Compact Framework[17], který nabízí řešení tohoto problému.

6.2. Ditrect3D Mobile

K získání zkušeností s Direct3D Mobile jsem přečet několik publikací[18]. Většina knih a tutoriálu na internetu se věnuje desktopové verzi Direct3D a nejčastěji využívají programovacího jazyka C++. Tyto knihy mi pomohly k pochopení základů a jakým způsobem všechno v Direct3D funguje. Přechod mezi jazyky C++ a C# není velkým problémem, protože pojmenování tříd a metod je ve většině případů totožné a princip zůstává stejný. Jako úvod do Direct3D Mobile mi posloužily vzorové příklady, které jsou součástí Windows Mobile SDK a další pak lze nalézt na webu společnosti Microsoft[19]. Příkladů je několik a každý z nich demonstruje určité schopnosti a vlastnosti Direct3D Mobile.
Nezmiňuji zde popis celé knihovny a všech jejich součástí, ale jen důležité částí a především oblast, která byla využita při implementaci aplikace.

Device

Tato třída umožní přímé spojení s grafickým adaptérem. Microsoft řeší přístup ke grafickému adaptéru právě přes tuto jednu třídu. Taktéž je řešeno i ovládaní zvuku nebo herního vstupního zařízení.

[17] Článek o lokalizaci, URL: http://www.codeproject.com/KB/mobile/UILocalizationWithCF20.aspx
[18] Publikace jsou uvedeny ve zdrojích [2], [3] a [4].
[19] D3D Mobile, URL: http://msdn.microsoft.com/en-us/library/ms181014%28VS.80%29.aspx

Před vytvořením instance *Device* je třeba inicializovat okno (*Form*), ve kterém se vše bude zobrazovat. Po inicializaci okna je třeba nastavit parametry pro vykreslování. Třída *PresentParameters* slouží k uchování zvoleného nastavení, které bude později předáno při inicializaci *Device*. Lze zde nastavit tyto parametry: Počet zadních bufferu (*BackBufferCount*) lze nastavit od 1 až po 3, záleží na jednotlivých zařízeních, kolik jich podporují. Formátem bufferu (*BackBufferFormat*) je myšlena jeho bitová hloubka (16 nebo 32bit). Lze nastavit i velikost zadního bufferu (*BackBufferWidth*, *BackBufferHeight*). Způsob jakým se budou buffery vyměňovat (*SwapEffect*), zde je nejpoužívanější metoda *Discart*. Nastavení okna pro vykreslování (*Windowed*), při hodnotě nepravda se použije šíře a výše okna podle zadaného bufferu, jinak dle okna (*Form*), ve kterém je scéna zobrazována. Po zadání těchto parametrů se může přistoupit k vytvoření instance *Device*. Při inicializaci se předává číslo grafického harwarového adaptéru, většinou 0. Pokud však zařízení obsahuje více adapterů, je zde možnost si zvolit, který bude využíván. Dále je nutné předat odkaz na okno (*Form*), ve kterém se vše bude vykreslovat a poslední důležitou částí jsou parametry, vytvořené v předešlém kroku. V této fázi je vytvořené spojení s grafickým adapterem. V poslední řadě je třeba se postarat o událost, která nastává při minimalizaci okna. Tato událost je důležitá proto, že dochází ke ztrátě spojení na grafický adaptér a po opětovné maximalizaci je třeba jej zpět navázat.

Překreslení displeje se provádí pomocí metody *OnPaint*, kterou je třeba přepsat. Samotné vykreslování probíhá v bloku, který začíná *BeginScene* a končí *EndScene*. Na konci tohoto bloku však ještě nedochází k předání údajů na displej. Je třeba použít metodu *Present*, která údaje předá dále až po zobrazení na displej. Podmět k překreslení obrazovky dává metoda *Invalidate*, jejíž volání značí, že údaje na obrazovce už nadále nejsou platné a je třeba nového vykreslení. Aplikace jako jsou hry nebo animace, kde jsou objekty pořád v pohybu využívají smyčky překreslování. Směrodatné zde je, kolik je aplikace schopna zobrazit snímků za sekundu (FPS – Frame per Second), aby se zajistila plynulost pohybu. U aplikací se statickým obsahem, který se tak často nemění se vyplatí překreslovat jen v momentě nějaké změny ve scéně.

Blok vykreslování (*BeginScene*, *EndScene*) by měl vypadat asi takto: Prvním krokem je vyčištění obrazovky voláním metody *Device.Clear*, která vyplní obrazovku jedinou barvou. Druhým krokem je nastavení matic[20] a světel[21]. Nyní je všechno připraveno a může se začít s vykreslováním jednotlivých objektů.

Poslední nastavení se provádí nastavováním parametrů ve třídě *RenderState*. Uvádím zde jen parametry, který jsem sám nastavoval. Osvětlení (*Lighting*) může být zapnuto (k osvětlení se využívá definovaných světel) nebo vypnuto (scéna je rovnoměrně osvětlena a definované světla se neberou v potaz). Je třeba uvést do chodu *ZBuffer*[22].

CullMode určuje způsob zanedbání polygonu, které jsou zády ke scéně. Většinou se nastavuje na hodnotu *Cull.CounterClockwise*, nebo na hodnotu *Cull.None*, která se využívá hlavně při testování aplikace.

[20] Druhy matic a jejich použití viz. kapitola 6.2 Direct3D Mobile - Pipeline

[21] Nastavení světel viz. kapitola 6.2 Direct3D Mobile - Lights

[22] Vysvětlení ZBuffer viz. kapitola 5.10 Depth Buffer

Vector

Třída vektor je v Direct3D Mobile obsažena ve třech variantách: *Vector2*, *Vector3*, *Vector4*. Číslo na konci každého z názvu značí, kolik má daný vektor rozměrů (počet souřadnic). Hlavní využití vektorů není tvorba 3D objektů, ale asistence při tvorbě matic. V běžné aplikaci se 3D scéna nevytváří pomocí vektorů, ale importuje jako celek (*Mesh*). Modelovaní definováním jednotlivých bodů a polygonů by bylo při složitější scéně nemožné. Třída *Vector* obsahuje funkce na určení normály, sčítaní, odečítaní, násobení vektorů a další.

Matrix

Třída, která usnadňuje vytváření a práci s maticemi. Jak již bylo zmíněno dříve[23] matice jsou u 3D aplikací důležitou součásti, protože se pomocí nich provádějí veškeré transformace objektů. Pro všechny operace jsou v této třídě funkce (rotace podle os, posuvy, zmenšování, zvětšování, násobení, invertní matice, určení determinantu). Na matice lze použít také operátory (+, -, *, ==, !=). Nově vytvořená matice obsahuje samé nuly. Při tvorbě matice, která provede například rotaci kolem jedné z os, je nutné vždy vycházet z matice jednotkové (*Matrix.Identity*).
Součástí třídy jsou dvě speciální matice: *Perspective*, *LookAt*[24].

Struktury 3D objektu

Direct3D Mobile definuje několik možností pro ukládání objektů, ze kterých se pak následně dají také vykreslovat.

CustomVertex – Základní jednotka. Nese v sobě informace o jednom bodu (vertexu).
Typů vertexů je několik a lze je rozdělit na skupiny. První skupina udává pozici vektorů v prostoru (*Position*) a druhá již převedené souřadnice (*Transformed*), podle kterých se přímo zobrazuje na displeji (neuplatňuje se na nich žádna transformace ani osvětlení). U obou skupin pak je ještě několik variant, které jsou sdílené. Vytvořit lze obyčejný vertex, obarvený (*Colored*), s texturou (*Textured*) nebo obarvený s texturou. U vertexu *Position* si lze ještě vybrat, jestli bude udávat i normálu vertexu (*Normal*). Například může typ vertexu vypadat takto: *CustomVertex.PositionNormalTextured.*
Transformed vertexy se využívají hlavně pro zobrazování 2D objektů ve 3D prostoru. Můžou jimi být různé informační či navigační prvky aplikace.

VertexBuffer - Ve své podstatě je to vlastně pole s vertexy. Při inicializaci je třeba určit, jakého typu jsou obsažené vertexy, jejich počet, odkaz na *Device* a specifikovat umístění v paměti, které lze zvolit z těchto tři možnosti: *Pool.Managed* (systém sám vybere úložiště), *Pool.SystemMemory* (systémová paměť), *Pool.VideoMemory* (paměť grafické karty). Naplnění bufferu se může provést dvěmi způsoby. První je použít metodu *SetData*, té vše

[23] Použití matic viz. kapitola 5.3 Transformace
[24] Popis matic viz. kapitola 6.2 Direct3D - Pipeline

předáme a data (většinou vertexy uložené v poli) se zkopírují do bufferu. Druhá možnost je komplikovanější. Je třeba zamknout buffer a data nahrát přímým přístupem do paměti pomocí třídy *GraphicsStream.* Po dokončení operace je nutné buffer odemknout.

IntexBuffer – Slouží k tvorbě polygonu pomocí indexování. Vyžaduje použití zároveň s *VertexBuffer*. IndexBuffer obsahuje sekvenci odkazů na vertexy ve *VertexBuffer*, pomocí kterých se postupně vytvářejí polygony. Samostatně jej použít nelze.

Mesh – Nejsložitější struktura, která v sobě obsahuje obě předešlé (*VertexBuffer* a *IndexBuffer*). Direct3D obsahuje několik vzorových *Mesh*. K dispozici jsou objekty jako krychle, koule, jehlan, kužel a válec. Chybí zde slavný šálek čaje (Tea pot), který je obsažen v desktopové verzi. Těchto objektů se většinou využívá jen u demonstrativních příkladů.

Při vykreslování z *VertexBuffer* jsou dvě možnosti. Lze použít metodu *Device.DrawPrimitives*, která se dá použít na *VertexBuffer* samotný nebo v kombinaci s *IndexBuffer*. Ještě předtím, než je použita metoda pro vykreslování, je důležité nastavit zdrojový *VertexBuffer* pomocí metody *Device.SetStreamSource*. V případě, že jsou použity také textury, musí se i ty nastavit (*Device.SetTexture*). Při použití kombinace s *IndexBuffer* je třeba jej zadat také. Metodě *Device.DrawPrimitives* se předá způsob, jakým se budou polygony vytvářet (*TriangleList*, *TriagnleFan*, atd.[25]), počáteční vertex a jejich celkový počet k vykreslení.
Vykreslování ze struktury *Mesh* se provádí přímo metodou *DrawSubset*, kterou třída *Mesh* obsahuje.

Font

Třídu lze nalézt jak v *System.Drawing.Font*, tak i v *WindowsMobile.DirectX.Direct3D.Font*. Jejich funkce je však rozdílná. Třída nacházející se v Direct3D přebírá při vytvoření třídu *Font* náležící *Drawing* a nabízí pouze metodu pro vykreslení tohoto písma (*DrawText*), na specifikované pozici. Veškeré nastavení vlastností písma (druh, velikost, styl, barva) je nutné provést ve třídě *System.Drawing.Font*.

Lights

Mobilní Direct3D poskytuje tři typy světla (Directional, Point a Ambient). Popis vlastností, které se dají nastavit je uveden níže. U Directional a Point světel lze nastavit barvy všech tři složek světla (Ambient, Diffuse, Specular)[26].
Directional – Zdroj světla bez pozice se zvoleným směrem. Nastavuje se pouze vektor *Vector3*, který nám určuje směr záření světla.
Point – Světlo vycházející z jednoho zdroje všemi směry. U tohoto světla se nastavuje jeho poloha, ta je určena vektorem *Vector3*. Dalším parametrem je dosah světla (*Range*), který značí hranici, za kterou už se světlo ve výpočtech osvětlení scény nebere v potaz. Světlo

[25] Způsoby tvorby polygonů viz. kapitola 5.2 3D objekty.
[26] Složky světla viz. kapitola 5.5 Osvětlení

umožňuje také nastavit křivku s jakou bude klesat intenzita světla. Výpočty se provádějí pomoci tohoto vzorce 1 / (att0 + att1 * d + att2 * d^2). Kde d zastupuje vzdálenost mezi objektem a světlem, att0 až att2 pak konstanty, které lze nastavit.
Ambient – Osvětluje celou scénu a každou část všech objektů rovnoměrně jednou barvou. Direct3D poskytuje jedno nezávislé ambient osvětlení, které se nevztahuje k žádnému použitému světlu. Lze jej tedy použít i samostatně. Nastavení tohoto světla lze provést v *Device.RenderState.Ambient*.

K vytvoření světla dochází tímto způsobem: Třída *Device* obsahuje pole *Lights*[*int*]. V tomto poli se musí na zvolené pozici nastavit parametry jednotlivých světel. Maximální počet aktivních světel je různý a záleží na možnostech použitého zařízení (toto lze zjistit pomocí *Caps.MaxActiveLights*). Vyžaduje-li scéna větší počet světel, než je maximální počet nějakého zařízení, pak je jedinou možností, vždy mít aktivní světla jen ta, která jsou v daném pohledu viditelná. Jeden z parametrů světla je stav světla, kde je třeba jej uvést do polohy zapnuto. Implicitně jsou všechna světla vypnuta. Dále je potřeba se ujistit, že je zapnuté osvětlení v *RenderState.Lightting*.
Osvětlení je v mobilní verzi Direct3D ochuzeno o druh světla Spot. Tento druh se využívá nejméně a lze s ním simulovat třeba světlomety automobilu, nebo světlo příruční baterky. Světlo z tohoto zdroje je vyzařováno v podobě kužele.

MeshLoader

V pořadí již druhá třída, která není oficiální součástí .NET Compact Framework ani DirectX Mobile. Třídu lze najít v *Microsoft.Samples.MD3DM*. Třída slouží k načítání 3D modelů do struktury *Mesh*. Desktopová verze Direct3D disponuje přímou podporou pro formát X file, ze kterého se dají modely načítat. Mobilní verze nemá tento import implementován a nabízí proto jiné řešení. Třída *MeshLoader* pracuje s binárním souborem typu MD3DM. Součástí ukázky od společnosti Microsoft je také konvertor *MeshConverter*[27], který provádí konverzi ze z X file souboru do MD3DM. S použitím tohoto formátu dochází také k úspoře místa, protože binární soubor zabere mnohem méně místa na disku.

X File

Tento formát je hlavním využívaným u desktopové verze DirectX a slouží k ukládání 3D modelů (*Mesh*). Uvádím jej zde proto, že je využit při převodu do formátu MD3DM, který lze mobilní aplikací načíst. Formát byl vytvořen společností Microsoft a jedná se o volný formát (nevztahuje se na něj žádná licence). Existuje ve dvou verzích, binární a textová, která je mnohem více využívána. Export do X file je podporován také 3D modelovacími nástroji (např. AutoCAD, 3D MAX, MAYA, atd.). Prohlížení X file lze provádět pomocí programu DirectX Viewer, který je součástí desktopové verze DirectX SDK. Na stránkách[28] společnosti Microsoft lze nalézt referenční příručku k tomuto formátu.

[27] MeshConverter [CD], soubor: Application/MeshConveter/MeshConverterCS.exe
[28] X File formát, URL: http://msdn.microsoft.com/en-us/library/bb173014%28VS.85%29.aspx

Pipeline

Způsob jakým se všechny body převádějí z 3D prostoru na 2D, tak aby je bylo možné zobrazit na monitoru, či displeji se nazývá pipeline. Na každý bod se aplikuje sada matic, kterou je potřeba před začátkem vykreslování scény zadat.
První z nich nese název world (matice světa). Na této matici lze provádět posuny, rotace atd.
Při vykreslování jsou jednotlivé *Mesh* vždy umístěny do středu souřadnicového systému. Matice world pomáhá umisťovat jednotlivé *Mesh* na požadované pozice.
Další v pořadí je matice s názvem view (umístění kamery), ve které se nastavuje pozice kamery, směr kterým se kamera dívá a její natočení. V Direct3D lze tuto speciální matici vytvořit pomocí metody *Matrix.LookAt*.
Poslední maticí je projection (nastavení projekce). V této diplomové prácí se využívá výhradně perspektivní projekce[29]. Nastavuje se zde úhel záběru kamery, zkreslení, dále pak rozsah kamery (*clipping*). K rozsahu jsou přiděleny dvě hodnoty reprezentující interval vzdálenosti pro vykreslování, vše co je před nebo za není vykresleno. Úhel záběru se standardně nastavuje na hodnotu 45 stupňů (pi/4). Zadává se v radiánech. Jeho zmenšováním se může docílit zoomování ve scéně. V případě, že úhlem přesáhne hranici 45 stupňů, bude docházet ke zkreslení a scéna bude vypadat jako při použití rybího oka. Matice projection se dá v Direct3D vytvořit pomocí metody *Matrix.Perspective*.

[29] Perspektivní projekce viz kapitola 5.11Projekce

7. Tvorba 3D mapy

Jako vzorová oblast pro modelování byla vybrána ostravská zoologická zahrada. Vzhledem k rozloze areálu a jeho komplikovanosti se nebude vytvářet kompletní 3D mapa areálu doplněná o všechny informace o zvířatech, ale jen její malá část, která poslouží jako demonstrační příklad, jakým způsobem lze mapu vytvořit. K modelování jsem zvolil pavilón slonů a jeho nejbližší okolí.

7.1. Zdroje GPS dat

Vzhledem k časové náročnosti metody sběru vlastních dat[30], jsem se rozhodl tuto metodu netestovat. Data jsou tedy získána z obrazových map a jiných zdrojů na internetu.

7.1.1. Data ze serveru OpenStreet Map

Prvním dobrým zdrojem dat je tento server. Všechny informace jsou volně šiřitelné a jsou zde nahrávány uživateli. Sever používá systém WGS a tudíž je nutná i konverze dat do systému S-JTSK. Zvolená lokalita ZOO je zde zmapována celá. Obr. 7.1.1.a znázorňuje cesty a jednotlivé části ZOO získané z tohoto serveru.
Přímo z webu lze stáhnout data v těchto formátech: PNG, JPG, SVG, PDF, HML a OSM. Z tohoto výčtu se k další úpravě dají využít tyto formáty: SVG a OSM. V případě, že jsou

Obr. 7.1.1.a – Mapa Ostravské ZOO (OpenStreetMap)

[30] Metoda sběru vlastních dat viz. kapitola 3.5.1 Sběr dat v terénu

data stáhnuta ve formátu SVG[31], dojde přímo k převodu do formátu vektorové grafiky. Tento export by byl vhodný za předpokladu, že se i nadále pracuje se systémem WGS. Při exportu dojde k převodu dat (ze stupňů na decimální jednotky s ohledem na měřítko a velikost exportované mapy). Tím by se značně zkomplikoval i jejich převod. Druhým využitelný formát je OSM, který je napsán pomocí XML. Pro snadnější přístup a manipulaci s daty je dobré si stáhnout editor, který tento sever zdarma poskytuje. Jedná se o aplikaci napsanou v jazyce Java s názvem JOSM. Pomocí této aplikace se lze přímo připojit k serveru a stáhnout potřebná data. Aplikace využívá pouze datového formátu .OSM. Struktura uchovává data v systému WGS (decimální zápis). Jednotlivé pozice bodů jsou uloženy v elementu s názvem node, ve kterých je uložena pozice ve WGS, unikátní číslo záznamu a jméno uživatele, který data uložil. Dalším důležitým elementem je way (cesta), která obsahuje unikátní čísla jednotlivých node, a tak z nich vytváří cestu.

Z celého souboru je potřeba vybrat jen cesty, které jsou potřeba a ty převést do systému S-JTSK. Za tímto účelem jsem napsal konvertor, který data ze souboru vytáhne, převede a uloží do formátu SVG. Program má název OSMParser a lze jej nalézt na přiloženém CD. Na vstup programu je třeba předat zdrojový soubor (osm formát). Dále unikátní čísla cest, které mají být zahrnuty, měřítko výsledné mapy a cílový soubor (SVG formát). Možný výsledek je vidět na obr. 7.1.1.b, který obsahuje cesty procházející celou ZOO. V příloze A je uveden výpis z tohoto SVG souboru.

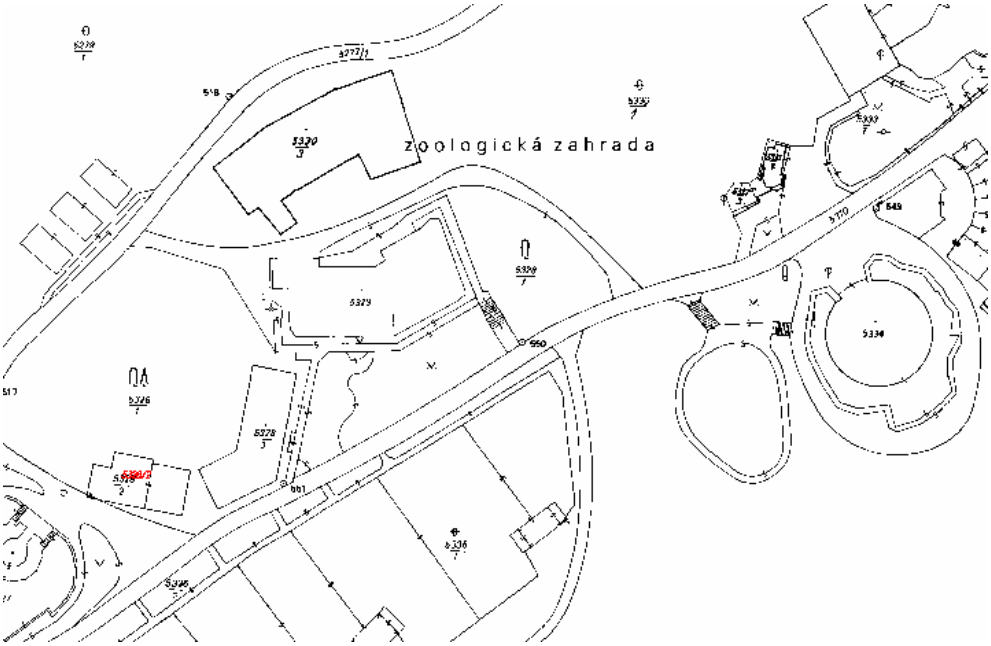
Obr. 7.1.1.b – Mapa cest Ostravské ZOO

Takto převedená mapa se dále může importovat do 3D grafického editoru a může se na ní začít budovat 3D mapa. Nevýhodou této metody je, že obdržené informace nejsou dostatečně detailní (šířka cest, atd.). Jako samotný zdroj nemůžou tyto informace posloužit k tvorbě 3D mapy.

[31] SVG formát viz. kapitola 3.4 Zápis a zpracování GPS dat

7.1.2. Mapy z katastru ČR

Jedná-li se o území České republiky je zde možnost využití internetových stránek Českého úřadu zeměměřičského a katastrálního (www.cuzk.cz). Katastrální mapy České republiky jsou uloženy v systému S-JTSK. Online prohlížení map katastru neumožňuje žádný export. Mapu lze získat použitím klávesy PrintScreen nebo pomocí WMS služby, kterou server



Obr. 7.1.2 – Mapa z katastrálního úřadu ČR

poskytuje. Její zobrazení je tak možné například v GoogleEarth[32] a poté její následný export jako obrázek. Příklad takto vyexportované mapy je uveden na obr. 7.1.2.
Při porovnání s předešlou metodou bylo získáno mnohem více informací, které by se jinak musely získat vlastním měřením v terénu. Jsou zde půdorysy veškerých budov a tvary cest. Mapa jako obrázek se pak dá použít v 3D modelovacím programu jako podklad, na kterém lze 3D mapu vybudovat. Je důležité přesně určit měřítko mapy a umístění do souřadnicového systému modelovacího programu, aby později bylo možno na mapě navigovat.

Pro tvorbu mapy je lepší využít mapy z katastru, protože usnadní práci při tvorbě 3D modelu. Převedených map ze serveru OpenStreet Map se dá využit k porovnání a přesnějšímu určení jednotlivých pozic na mapě[33]. U obou těchto variant chybí údaj nadmořské výšky. Proto v této práci došlo k jeho zanedbání, a to z několika důvodů. K získání tohoto údaje by bylo potřeba provést vlastní měření. Tvorba 3D mapy by se tím značně zkomplikovala. Poslední překážkou by pak bylo znázorňování pozice na mobilním zařízení. K přesnému určení nadmořské výšky je zapotřebí vyššího počtu satelitů ve výhledu. Často tak dochází k tomu, že údaj nadmořské výšky je značně zkreslený.

[32] Nastavení služby, URL: http://www.civil3d.cz/2009/09/lze-ulozit-wms-katastralni-mapy.html
[33] Rozdíly mezi údaji v mapách viz. kapitola 9.3 Test v terénu

Úplně ideálním případem by bylo získat stavební plány lokality. Z těchto materiálu by se dal model vytvořit s vysokou přesností. Musí se však počítat také s tím, že modelování by pak zabralo mnohem více času.

7.2. Tvorba modelu

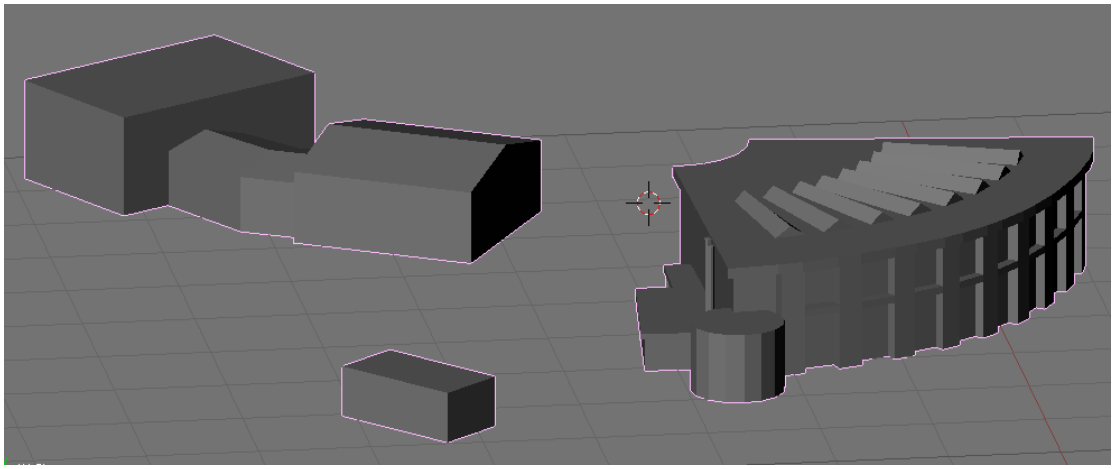
Veškeré modelování a mapování textur proběhlo v programu Blender[34]. Vytváření a úpravy textur pak v programu Gimp, sloužícím pro editaci rastrové grafiky. Tento software podléhá GNU licenci. Podklady pro tvorbu modelů budou tvořit převážně fotografie pořízené v areálu ZOO a GoogleMaps. GoogleMaps disponují funkci StreetView, která umožňuje přímo procházení po vybrané ulici, či cestě. Jedná se o pseudo 3D prostředí vytvořené pomocí fotek. Areál Ostravské ZOO je tímto způsobem zmapován takřka celý. Využiji tedy i tento zdroj. Protože nejsou k dispozici žádné stavební plány lokality, ani jednotlivých budov, bude tvorba modelu probíhat pomocí plánu z katastrálního úřadu a pořízených fotografií. Vertikální vzdálenosti tedy nebudou přesné. Model slouží pouze jako grafický prvek, který je zobrazován uživateli. Na první pohled by mělo byt uživateli jasné, ve které části modelu se v realitě nachází. Skutečnost, že vertikální informace bude zkreslená nijak neovlivní používání aplikace a její funkčnost. Horizontální vzdálenosti budou přebrány z mapy katastru a budou odpovídat skutečnosti. Veškeré textury budou vytvořeny z pořízených fotografií a obrázků z GoogleMaps.
Při tvorbě mapy je potřeba postupovat systematicky. Celou scénu je třeba rozdělit do několika jednotných celků, které pak při pozdějším sestavení scénu utvoří. Jednotlivé části budou přehlednější a snadněji editovatelné. Pro aplikaci běžící na mobilním zařízení by bylo výhodné umožnit snížení počtu objektů (polygonů) k zobrazení. Jednou z možností je rozdělit celý model do několika podskupin. Například takto: základ mapy (cesty, krajina, půdorysy budov), budovy, různé další detaily (lavičky, značky, keře, atd.). Uživatel si pak může zvolit, které vrstvy se budou vykreslovat a které ne. Tím se zajistí chod také na zařízeních se slabším výpočetním výkonem.
Je zde zmíněn pouze zobecněný postup, kterým lze postupovat s využitím kteréhokoli 3D modelovacího nástroje. Pro detailnější postupy a informace o aplikaci Blender proto doporučuji knihu Blender For Dummies [1] a tutoriály na stránce www.blender.org.

7.2.1. Modelování

Postup modelování může být různý. Celý model se může tvořit na jedné scéně, a rozdělit později, nebo rozdělit ihned. Prvním krokem je vložení mapy v obrázkovém formátu, získané z katastrálního úřadu, jako pozadí (Background Image). Na něm pak lze začít tvořit model. Po vytvoření jednoduché kostry se postupně přidávají detaily. Cely proces je časově náročný a závisí hodně na zkušenostech s 3D modelováním, a také na zkušenostech s modelovacím programem.

[34] Popis programu viz. Kapitola 4.2 Blender

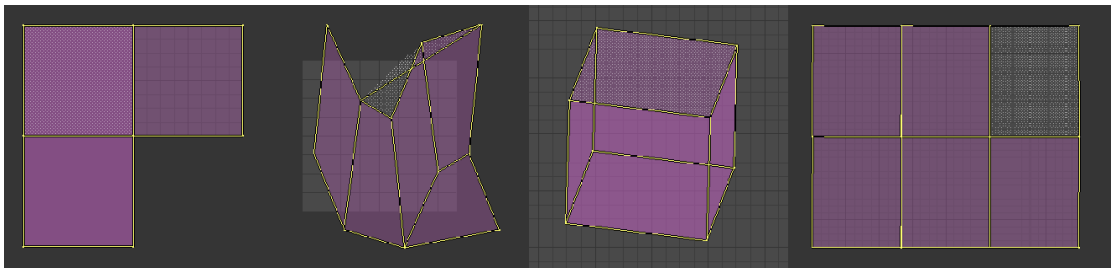
Vytvořený model si lze prohlédnout na obr. 7.2.1. Na mapě z katastrálního úřadu jsem nejprve vytvořil hrubé obrysy půdorysů budov a cest. V dalším kroku byl model rozdělen na několik částí, u kterých pak probíhalo modelování samostatně.



Obr. 7.2.1 – 3D Model – Pavilón slonů (bez textury)

7.2.2. Mapování textur

Mapování textur je zdlouhavý proces. Správným postupem je, že po vytvoření modelu lze využít funkci Blendru, které texturu rozbalí do 2D prostoru na obrázek. Ten se pak v libovolném rastrovém editoru obarví a textura je hotová. Tato metoda se nazývá UVmapping. Blender nabízí několik způsobů, jak 3D objekt převést do 2D. Jakým způsobem jednotlivé převody pracují je patrné z obr. 7.2.2.a, kde je provedena demonstrace na krychli. První část je mapa textury rozbalená pomocí krychlové projekce, dále cylindrická projekce. Předposlední mapa je provedena pomocí momentálního pohledu na scénu a poslední využívá smart projection, která se vždy snaží o co nejlepší rozmístění všech prvků s co nejmenším zkreslením. Využití jednotlivých způsobů se dá kombinovat, když se bude 3D objekt rozbalovat po částech. Takto rozbalený objekt se exportuje do obrázkového formátu a obarví v rastrovém editoru za použití fotografií. Pro pořizování fotografií je ideální dobré počasí a v tomto případě také minimální počet návštěvníků ZOO.



Obr. 7.2.2.a – Uvmapping krychle

Další možností je vytvořit texturu a tu postupně mapovat na jednotlivé polygony 3D objektů. Tato možnost se obecně moc nevyužívá. Mapování je zdlouhavé. Protože model bude

zobrazován na mobilním zařízení, je zde snaha redukovat velikost použitých textur. Došlo proto ke kombinaci dvou zmíněných metod pro tvorbu textury. Části 3D objektu, které jsou si ve skutečnosti hodně podobné nebo stejné jsou ve 2D obrázku mapovány vždy na stejný úsek textury. Budova tvaru kvádru, která by měla všechny strany totožné, tak zabere čtvrtinu místa v paměti. Vzniklý prostor se může zaplnit texturou s většími detaily nebo ušetřit. Souborů s texturami by mělo být co nejméně. Na každý *Mesh* proto náleží jen jeden soubor s texturou. Výsledný model budov s texturou je na obr. 7.2.2.b a obrázek textury byl zanesen do přílohy B.

Obr. 7.2.2.b - 3D Model – Pavilón slonů (s texturou)

7.3. Export

V momentě, kdy je model hotový, je třeba jej převést do formátu zobrazitelném pod DirectX mobilní verze. Blender ukládá vše do vlastních souborů s příponou .blend. Blender umožňuje export do X file a z tohoto formátu se pak pomocí konvertoru *MeshConverter*[35] dostane do cílového .md3dm formátu, který bude aplikace používat.

7.3.1. .blent do .x

Při tomto exportu je třeba nastavit v aplikaci Blender následující: Volba s názvem “Flip z“ musí být aktivní. Toto je z důvodu, že se přechází z pravoruké do levoruké soustavy souřadnic[36]. Jako další je třeba označit volbu s názvem “Swap zy“. Tímto příkazem se

[35] MeshConverter viz. kapitola 6.2 Direct3D Mobile – MeshLoader

[36] Levoruká soustava viz. kapitola 5.1 Koordinační systémy 2D a 3D

prohazují osy z a y. Blender používá systém souřadnic, který má osu z ve svislém směru. Posledním nastavením je volba “no smooth“, která zajistí, že se nebudou přepočítávat normály modelu. Výsledek operace si je možné prohlídnout otevřením souboru v aplikaci DirectX Viewer. Po převodu je třeba se ujistit, že je ke každému materiálu přiřazená nějaká textura. Soubor se dá otevřít v libovolném textovém editoru a je třeba provést kontrolu úseku MeshMaterialList, kde každý Material musí mít přiřazenu hodnotu TextureFilename. Pokud tuto položku neobsahuje, stačí ji překopírovat ze sousedního materiálu a výsledek uložit. V případě neodstranění tohoto nedostatku dojde k chybě při načítání souboru do aplikace.

7.3.2. .x do .md3dm

Tento převod v sobě nezahrnuje žádné nastavování. Stačí využít programu *MeshConverter*, kterému se jako parametr předává vstupní a výstup soubor. Soubory s texturami nemusí být součástí složky při převodu, na nich se žádná konverze neprovádí. Výsledný soubor se pak načítá přímo pomocí třídy *MeshLoader*[37].

[37] Popis třídy MeshLoader viz. kapitola 6.2 Direct3D Mobile - MeshLoader

8. Tvorba Aplikace

8.1. Specifikace požadavků

Aplikace poběží na platformě Windows Mobile 6 na zařízení se zabudovaným GPS přijímačem a dotykovým displejem. Aplikace bude zobrazovat vytvořenou 3D mapu zvolené části lokality a navigovat na ní pomocí zabudovaného GPS přijímače. K jednotlivým objektům (budovám) budou přístupné podrobné informace.

8.2. Vlastní specifikace požadavků

Na základě specifikace požadavků jsem si přesněji určil několik vlastních požadavků, které by aplikace měla splňovat.

- Struktura menu aplikace by měla byt jednoduchá.
- Aplikace by měla být schopna přímo zobrazit pouze GPS informace přijímané ze satelitu, nezávisle na mapě. Při použití s 3D mapou by měla zobrazovat aktuální pozici na 3D mapě. Přijímání GPS informací ze satelitu by mělo být možné kdykoli spustit nebo zastavit.
- Zařízení, které nedisponuje GPS přijímačem by mělo být schopno využít této aplikace bez možnosti navigování.
- Zobrazovaná 3D mapa bude ve formátu MD3DM, uživateli by mělo být umožněno pohybovat se po mapě, naklánět pohled na mapu a přibližovat/oddalovat ji.
- Podrobné informace o objektech se budou uživateli prezentovat seznamem, ze kterého si bude moci nějaký vybrat a zobrazit jeho podrobné informace, nebo zobrazit polohu objektu přímo na 3D mapě.
- Informace o objektech budou stránky s HTML obsahem, které můžou byt doplněny o styly (CSS).
- Aplikace bude data přijímat importem z XML souboru. XML souboru bude obsahovat základní informace o mapě, odkazy na soubory s 3D modely (ve formátu MD3DM), odkazy na soubory s informacemi o jednotlivých částech lokality (formát HTML).

8.3. Použitý software

Při implementaci a testování aplikace jsem využil níže uvedený software. Nejsou zde zmíněny aplikace, které posloužily k úpravě GPS dat a tvorbě 3D modelu.

- Microsoft Visual Studio .NET 2008 – Vývojové prostředí.

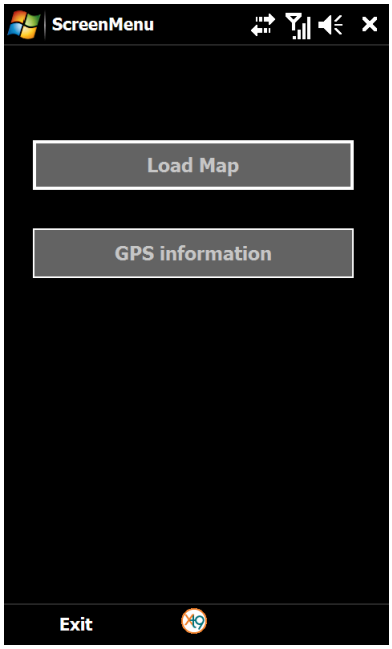
- Microsoft Windows Mobile 6 SDK – Knihovny pro mobilní zařízení.
- .NET Reflector – Průzkumník pro knihovny .NET
- Microsoft ActiveSync – Synchronizace mezi PC a mobilním zařízením.
- QuickCab 2.0 – Vytváření instalačních balíčku pro mobilní zařízení.

8.4. Popis jednotlivých tříd

Všechny třídy, které dědí ze třídy *Form* jsou vždy na počátku názvu označeny slovem screen. Označením všech tříd, které slouží jako zobrazovací jednotky, tak došlo ke zlepšení přehlednosti. V následující části práce je uveden popis jednotlivých tříd a jejich funkce v programu. Některé z nich jsou zde popsány detailněji než jiné. Důvodem toho je, že funkce některých tříd je důležitá a je vhodné si je popsat a vysvětlit. Dodatečné informace o všech proměnných a metodách lze nalézt v programátorské dokumentaci. Všechny třídy, které mají nějaký grafický výstup jsou u popisu doprovázeny obrázkem.

ScreenMenu

Start aplikace proběhne ve třídě Program a třída *ScreenMenu* je úplně první okno, které je uživateli zobrazeno. Jeho vzhled si lze prohlédnout na obr. 8.5.a. Jedná se o jednoduché rozhraní vytvořené pomocí návrhového pomocníka Visual Studia. Umožňuje načtení mapy z XML souboru (Load Map). Zobrazení samostatných GPS informací přijímaných ze satelitů (GPS information), nebo ukončení aplikace (Exit). Načítání mapy z XML souboru je doprovázeno zobrazením dialogového okna, ve kterém lze procházet úložiště zařízení a vybrat tak soubor. Dochází pouze k uložení cesty k souboru a řízení je dále předáno třídě *ScreenLoading*. Při zobrazení GPS informací dochází k inicializaci ovladače pro GPS přijímač a třídy *ScreenGPS*, která zobrazuje informace obdržené ze satelitů.



Obr. 8.5.a – Základní menu

Globals

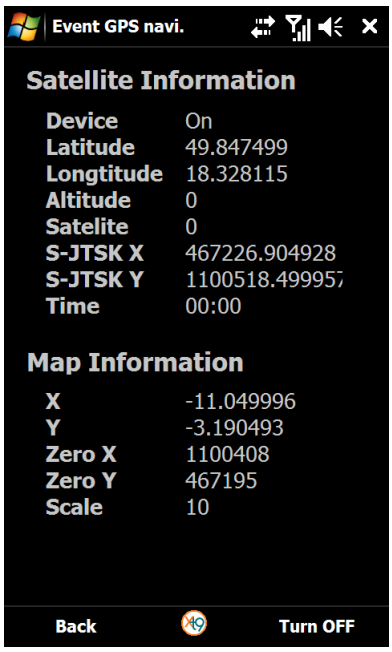
Třída slouží k udržení instancí, které jsou využívány na více místech aplikace a jejich předáváním by se tak značně zkomplikovala struktura aplikace. Je zde například uložena instance s přístupem k GPS přijímači. K tomuto prvku je přistupováno hned z několika částí aplikace, které využívají jejich funkcí.

GSPService

Je třída, která se stará o obsluhu GPS přijímače za pomocí balíčku *Microsoft.WindowsMobile.Samples.Location*. Po inicializaci je možné GPS přijímač zapínat *StartGPS*() a vypínat *StopGPS*(). Třída implementuje dvě události. První z nich *PositionChanged* nastává při změně GPS pozice. Jako argument předává již převedené souřadnice, se kterými přímo pracuje zobrazovaná mapa. Druhá událost *PositionChangedFullInfo* nastává ve stejném okamžiku a předává veškeré informace získané ze satelitu. Této události využívá okno *ScreanGPS*.
Součástí třídy jsou dvě statické metody, které slouží pro převod mezi souřadnicovými systémy WGS a S-JTSK. První z nich *WGStoSJTSKsimple* využívá algoritmu, který detailněji popsán v přiložených souborech[38] na CD. Tento algoritmus byl přepsán z tabulkového procesoru Excel a je součástí této metody. Obsahuje konverzi pouze v jednom směru, a to z WGS do S-JTSK. Druhá metoda *WGStoSJTSK* využívá složitějšího algoritmu s větší přesností, který je uložen ve třídě *GeoWGStoJTSK*. Tento soubor byl poskytnut vedoucím diplomové práce. V aplikaci se pro převod mezi soustavami používá výhradně toho algoritmu. Jeho výpočet je o jeden desetinný řád přesnější.

ScreenGPS

Třída pro zobrazování GPS informací. Může být zobrazena přímo z menu nebo až po načtení mapy ze souboru. Tato třída může zobrazovat pouze GPS informace ze satelitu nebo i informace o aktuálně používané 3D mapě (měřítko mapy, aktuální pozice). Informace o mapě lze zobrazovat nastavením proměnné *ShowMapInfo* typu *boolean*. Grafický výstup této třídy znázorňuje obr. 8.5.b.
V tomto okně lze vypínat a zapínat GPS přijímač. Tlačítko je v součástí spodního menu (Turn ON, Turn OFF).



Obr. 8.5.b – GPS informace

PointDouble

Struktura vytvořená z důsledku časté potřeby předávaní dvou hodnot typu Double. Třída *System.Drawing.Point* poskytuje předávání pouze dvou hodnoty typu *int*. Struktura obsahuje vlastnosti, pomocí kterých je možnost si hodnoty vrátit přímo v datovém typu *float*. Většina metod a tříd DirectX přebírá hodnoty právě v tomto formátu. Struktura má také jednu proměnnou *isValid* typu *boolean*, kterou se dá označit jsou-li data aktuální.

[38] Cesta k souboru /Texts/Transformace GPS.xls

PositionInfo

Třída, která nese informace získané ze satelitů a vypočtené hodnoty aktuální pozice. Obsahuje pozici v souřadnicích WGS, převedenou S-JTSK a vypočtenou souřadnici na zobrazované mapě. Dále je zde uložena nadmořská výška, počet satelitů ve výhledu, aktuální čas, nulové souřadnice mapy, na které se zobrazuje a její měřítko. Při vložení souřadnic ve WGS systému dochází k automatickému přepočtu a uložení na pozice souřadnic S-JTSK a souřadnic zobrazované mapy.

MapData

Tato třída obsahuje veškeré informace o mapě, která je vytvořena při načítání z XML souboru. Jsou zde jména souborů, ze kterých se načítají jednotlivé *Mesh* objekty, atd.
Třída je serializovatelná pomocí *System.Xml.Serialization.XmlSerializer* a obsahuje tři statické metody. První z nich *LoadMapDataSample* vrátí testovací data mapy (objekt *MapData* obsahující vzorovou mapu), která obsahují vzorovou mapu. Druhá *LoadMapDataFromXml* vrátí data, která načte z požadované lokace. Třetí *SaveMapDataToXml* data mapy přebírá a ukládá do XML souboru na zvolenou lokaci.
Jedinou nestatickou metodou této třídy je *CheckData*. Tato metoda provádí kontrolu načtených dat a jejich konzistenci. Je nutné předejít chybám, které by mohli později komplikovat zobrazení 3D mapy. Vzorový výpis z XML souboru s krátkým popisem je uveden v příloze C.

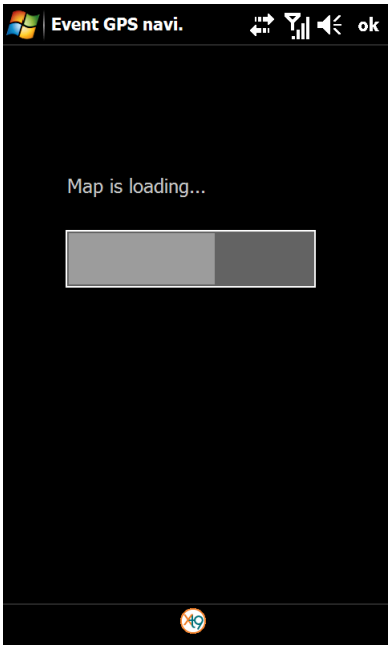
InstanceHolder

Při načítaní z XML souboru se provádí také inicializace veškerých komponent (*ScreenD3D*, *ScreenAttList*, *ScreenGPS*, atd.), které jsou dále uchovány v *InstanceHolder*. Třída zaniká až při ukončení práce s mapou a uvolňuje všechny zdroje. Tato třída obsahuje také několik metod, pomocí kterých se celá inicializace provádí po částech.

- *InitializeMapData* – Načtení z XML souboru a verifikace dat.
- *InitializeGPS* – Inicializace ovladače pro GPS přijímač.
- *InitializeD3D* – Inicializace DirectX zařízení.
- *InitializeOthers* – Inicializace všech ostatních oken a objektů.

ScreenLoading

Okno zobrazující stav načítání souboru. Tato třída pracuje s třídou *InstanceHolder*. Postupně volá její metody pro inicializaci jednotlivých částí a zobrazuje stav pomocí komponenty *ProgressBar*. Při vyskytnutí nějaké chyby během inicializace se zobrazí komponenta *MessgeBox* se zprávou pro uživatele. Zpráva říká, ve které časti inicializace se chyba vyskytla. Obr. 8.5.c znázorňuje toto okno i se zprávou o chybě při načítání. Po dokončení inicializace tato třída zaniká a řízení se předává ScreanD3D.



Obr. 8.5.c – Načítání ze souboru

Obr. 8.5.d – 3D mapa

ScreenD3D

Okno, které slouží pro zobrazování 3D obsahu pomocí DirectX. V této třídě se provádí veškeré vykreslování 3D mapy a jednotlivé výpočty, zejména matic, se dějí ve třídě *D3DControl*. Vzhled grafického výstupu si lze prohlednout na obr. 8.5.d. V horní části obrazovky se nachází komponenta *Label* zobrazující aktuálně vybranou atrakci. Prává část je vyhrazena přibližování a oddalování na 3D mapě. Tato část je vytvořena použitím vlastně naprogramované komponenty *MyButton* (viz níže). Spodní část obrazovky tvoří menu obsahující čtyři tlačítka. K u místění tlačítek bylo využito komponenty *Panel*, která tlačítka sdružuje a usnadňuje tak jejich umístění na obrazovce. Jednotlivá tlačítka jsou tvořena komponentou *MyButton* a jejich funkce je popsána níže. Spodní okrajová část obrazovky obsahuje menu, které umožňuje zobrazení oken *ScreenGPS* (veškeré GPS informace ze satelitů, GPS information) a *ScreenAttraction* (seznam dostupných atrakcí pro tuto mapu, Attraction List), vymazání zaměření na atrakci (Clear Focus), nebo ukončení práce s mapou a návrat do základního menu *ScreenMenu* (Close).

Manipulace s mapou má dva režimy. Přepínání mezi těmito režimy umožňují první dvě tlačítka spodního panelu. První slouží k posuvu mapy (ikona překřížených šipek) a druhý režim (ikona šipky ve tvaru půlkruhu) umožňuje rotaci mapy v horizontálním a vertikálním směru.Horizontální rotace není nijak omezena. U vertikální rotace došlo k omezení tak, že minimální úhel náklonu pohledu je 15 stupňů a maximální 90 stupňů. Popis výpočtů je uveden u popisu třídy *D3DControl*, kde také probíhá celý výpočet.
Dvě zbylá tlačítka slouží k přesunutí záběru kamery na aktuální GPS pozici nebo aktuálně zvolenou atrakci. K tomuto přesunu nedojde za předpokladu, že GPS přijímač není zapnut, nemá signál, nebo když ho zařízení vůbec nemá. U atrakce je pak nutné, aby byla nějaká vybrána v zaměření (její jméno se objeví v horní částí obrazovky v již zmíněné komponentě *Label*), jinak se po stisku tlačítka neprovede žádná operace.
Při vytvoření této třídy dochází k inicializaci postupným voláním těchto metod:

- *InitializeGraphics* – Dochází k vytvoření *Device*[39] a nastavení parametrů pro vykreslování scény (*Device.RenderState*).
- *initializeMatrix* – Vytvoření matic (world, view, projection, atd.). Výpočet matic probíhá ve třídě *D3DControl*.
- *initializeMeshes* – Načtení všech požadovaných *Mesh* (vrstvy mapy, ukazatele GPS, atd.) do paměti včetně textur.
- *initializeFormComponents* – Zde probíhá inicializace tlačítek a jiných komponent. Grafické rozhraní třídy ScreanD3D nebylo vytvořeno navrhovacího pomocníka Visual Studia. Je tedy nutné implementovat vlastní inicializaci.

Třída dále obsahuje sérii handlerů, které řeší události tlačítek. Každé z tlačítek reaguje na stisk. Manipulace s mapou (posuv a natáčení) je řešena samostatně ve třídě D3DControl.
Vykreslování probíhá v metodě *Render*. V prvním kroku překresluje celý obsah backBuffer jednou barvou. Vykreslování se vždy provádí do backBuffer z důvodu používání metody doubleBuffering[40]. V dalším kroku proběhne nastavení matic (world, view a projection) a světel. Nyní začíná vekreslování jednotlivých *Mesh*. Nejprve vrstvy 3D mapy, následovány ukazateli GPS polohy a ukazatele na aktuálně vybranou atrakci. Celý obsah je pak uvolněn k prezentaci uživateli voláním metody *Device.Present*.
Pro načítání jednotlivých Mesh obsahuje třída dvě statické metody. *LoadMeshFromResource* načítá *Mesh* přímo ze zdrojů aplikace (jsou součástí instalačního souboru aplikace). Této metody se používá pro načtení ukazatelů GPS pozice a aktuální atrakce. Druhá *LoadMeshFromFile* umožňuje načtení *Mesh* z externího souboru. Cesty a názvy souborů jsou součástí importovaného XML souboru. Obě tyto metody přebírají jako parametr odkazy na objekty (*Mesh*, *Material* a *Texture*), do kterých je načtení provedeno.

[39] Třída Device viz. kapitola 6.2 Direct3D Mobile - Device
[40] DoubleBuffering viz. kapitola 5.9 Back Buffers

MyButton

Tato třída vznikla ze dvou důvodů. Za prvé, třída *Button*, kterou .NetCompact Framework disponuje nepodporuje umístění obrázků na pozadí a druhým důvodem je, že tlačítka této třídy jsou schopna reagovat pouze na kliknutí myši, stisk, držení a uvolnění tlačítka klávesnice. Cílové zařízení by mělo byt vybaveno dotykovým displejem a pro to jsem se snažil držet pravidla, že veškeré ovládání je možné provést zmíněným displejem. Třída *MyButton* reaguje na všechny události, tak jako *Button*, a navíc obsahuje reakci na zmáčknutí, držení a uvolnění tlačítka myši, které jsou využity pro zoom.

D3DControl

V této třídě probíhají veškeré výpočty týkající se zobrazování 3D mapy třídou *ScreanD3D*. Je zde implementace události *MatrixChanged*, která se vyvolává v momentě, kdy dojde ke změně v jedné z matic (world, view, projection, gpsMatrix nebo attMatrix) a vznikne tak nutnost obrazovku překreslit. Uvádím zde postup, pomocí kterého se provádí výsledný výpočet matic při manipulaci s mapou.

Výpočet zoom

Přibližování a oddalování je prováděno pomocí matice projection (*Matrix.PerspectiveFovLH*) nastavením úhlu zorného pole (*fieldOfView*). Tento úhel by měl mít maximální hodnotu 45 stupňů, protože při použití větší úhlu by docházelo ke zkreslení 3D scény.
Nejdříve je třeba zjistit přesnou pozici kamery (*cameraPosition*), která se určí pomocí hodnoty maximálního zorného pole (*visibleWidthMaxSJtsk*) a měřítka mapy (*mapScale*). Tyto hodnoty lze zadat u parametrů mapy v XML souboru. Hodnota *visibleWidthMaxSJtsk* značí zorné pole v metrech, na kterou lze maximálně mapu oddálit. Kamera je umístěná na souřadnicích, kde x souřadnice je rovna nule a y se z jsou si rovny (*CameraCoordinates*). Nejdříve je nutné zjisti vzdálenost kamery od středu souřadnicového sytému pomocí vzorce 8.5.1.

$$camereDistance = \frac{\frac{\text{visibleWidthMaxSJtsk}}{2 * mapScale}}{projectionAngleMaxTan} \qquad (8.5.1)$$

V proměnné *projectionAngleMaxTan* je uložen tangent poloviny maximálního úhlu kamery, což je 22,5 stupňů. Souřadnice kamery pak vypočítáme ze vzdálenosti kamery (*cameraDistance*) podle Pythagorovy věty vztahem 8.5.2.

$$CameraCoordinates = \sqrt{\frac{cameraDistance^2}{2}} \qquad (8.5.2)$$

Při prvním zobrazení se poměrové číslo úhlu pohledu *zoomStart* určuje vztahem 8.5.3 pomocí hodnoty *visibleWidthStartSJtsk*. Hodnota udávaná v metrech opět značí velikost zorného pole. Stejným způsobem je pak určeno maximální přiblížení (*zoomMin*) a oddálení (*zoomMax*).

$$zoomStart = \frac{\frac{\text{visibleWidthStartSJtsk}}{2}}{cameraDisance} \tag{8.5.3}$$

Tyto hodnoty obsahují desetinné poměrové číslo, pomocí kterého dochází k výpočtu úhlu *fieldOfView* (předává se při tvorbě matice projection). Prvně je nutné vypočíst hodnotu *FieldOfViewRatio* vztahem 8.5.4, která nese hodnotu poměrového čísla poloviny aktuálního zorného pole.

$$FieldOfViewRatio = zoomStart * e^{zoom} \tag{8.5.4}$$

Proměnná *zoom* obsahuje hodnotu, která se pří přibližování a oddalování upravuje. Exponenciální funkce (e^{zoom}) byla přidána z důvodu, aby bylo přibližování a oddalování příjemné lidskému oku. Výsledný úhel *fieldOfView* se vypočte pomocí vztahu 8.5.5.

$$fieldOfView = \arctan(FieldOfViewRation) * 2 \tag{8.5.5}$$

Výpočet rotace

Rotace jsou provedeny na matici view. Horizontální rotace není nijak omezená a výsledná rotační matice vypadá takto: *Matrix.RotationX*(*rotationX+rotationXadd*). Kde *rotationXadd = x*rotationSensitivity*. Při přiložení prstu na obrazovku dojde k uložení této pozice (*mouseXpre* a *mouseYpre*). Při tažení prstem se pak postupně odečítá relativní posun vhledem k místu prvního dotyku (metoda *touchWatch*, která se spouští jako samostatné vlákno). Proto se v rotaci počítá se dvěmi hodnotami. Hodnota *rotationXadd* se mění při pohybu prstu na displeji a při jeho odtažení je přičtena k *rotationX* a vynulována.
Vertikální rotace probíhá úplně stejně, ale navíc má implementované omezení, které zabraňuje, aby úhel překročil stanovené hodnoty. Minimální uhel pohledu je 15 stupňů a maximální 90 stupňů.

Výpočet posuvu

Posuv se provádí manipulací s maticí world. Pří posuvu mapy se přebírají souřadnice x a y z displeje a určuje se jejich relativní posun vzhledem k místu prvního dotyku, tak jako při rotaci. Dále je vypočten krok *TranslateStep* podle vzorce 8.5.6 v závislosti na momentálním zorném poli *FieldOfViewRatio*.

$$TranslateStep = \frac{2 * FieldOfViewRatio * cameraDis\tan ce}{screanWidth} \qquad (8.5.6)$$

Výsledný vektor pro posun pak vypadá takto: *Vector3*(x**TranslateStep*, 0, y**TranslateStep*). Pomocí tohoto přepočtu se posun mapy chová přirozeně a místo dotyku kopíruje posuvem pohyb prstu po displeji. K menšímu zkreslení dochází ve spodní a horní části displeje v závislosti na náklonu pohledu

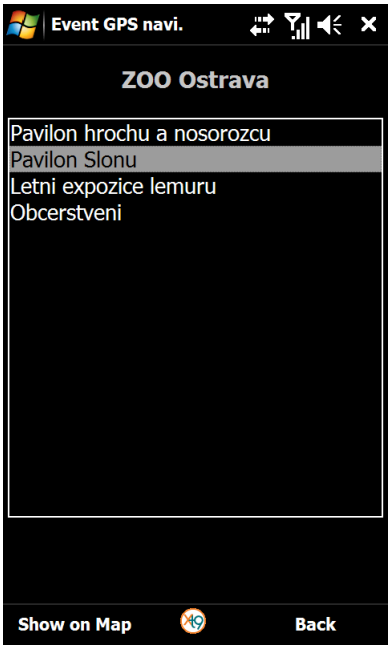
O výpočet výsledných matic se stará metoda *createMatrix*. K výpočtům s maticemi bylo využito funkcí třídy *Matrix* (rotace, posuvy, atd.).

Attraction

Třída, která nese informace o jediné atrakci obsahuje: Název atrakce (*attName*), její identifikační číslo (*attID*, přiřazeno při načítání ze souboru), cestu k souboru (*sourceFile*) obsahujícím detailní informace o atrakci. Poslední části je pozice atrakce uložená ve struktuře *AttractionLoacation*. U atrakce se vždy v XML souboru, ze kterého se načítá, udává její pozice ve WGS sytému. Po načtení se provede automatická konverze a pozice se uloží také v systému S-JTSK a souřadnicích 3D mapy.

ScreenAttList

Zobrazuje seznam dostupných atrakcí. Vzhled okna si lze prohlédnout na obr. 8.5.e. Seznam je tvořen komponentou *ListBox*. Pohyb po seznamu je jednoduchý a při dvoj dotyku se zobrazí detailní informace k dané atrakci (*ScreenAttDetail*). V horní části je zobrazen název



Obr. 8.5.e – Seznam atrakcí

Obr. 8.5.f – Detail atrakce

mapy pomocí komponenty *Label*. V menu umístěném ve spodní části jsou dvě volby. Pozici zvolené atrakce lze zobrazit na mapě (Show on map) nebo formulář opustit a vrátit se zpět (Back) k zobrazení 3D mapy.

ScreenAttDetails

Tato třída slouží k zobrazování HTML obsahu. Informace o jednotlivých atrakcích obsahují také obrázky. Text a obrázky jsou pro umístěny do HTML kódu a ke zobrazení se využívá komponenty WebBrowser. Jediným nedostatkem této komponenty je, že neumožňuje vypnutí horizontální posouvací lišty samostatně. Tento problém jsem vyřešil za pomocí HTML kódu. U komponenty WebBrowser jsou proto vypnuty obě lišty. HTML kód obsahuje komponentu (div), pomocí které se dá nastavit, aby zobrazovaná stránka měl pouze vertikální posuvnou lištu za předpokladu, že je jí potřeba. Předloha, jak by takový kód (HTML a CSS) měl vypadat je zahrnuta v příloze D. Přiklad jak výstup této třídy může vypadat je znázorněn na obr. 8.5.f.

9. Testování

9.1. Specifikace použitého zařízení

Aplikace byla navržena tak, aby byla spustitelná na jakémkoli zařízení s Windows Mobile verze 6 nebo 6.5. Zařízení musí podporovat knihovnu DirectX Mobile. Pokud mobilní zařízení nebude disponovat zabudovaným GPS přijímačem, nebude možné zobrazovat aktuální pozici na mapě. Zobrazení mapy a informací k jejím částem však zůstává plně funkční.
Veškeré testování proběhlo na zařízení Samsung Omnia II i8000. Níže jsou vypsány pouze parametry tohoto zařízení, které souvisejí s touto prací. Za nimi následují DirectX vlastnosti a omezení tohoto zařízení. Tyto informace jsou získány pomocí výpisu ze třídy *Microsoft.WindowsMobile.DirectX.Direct3D.Device.DeviceCaps*.

Obecná specifikace zařízení:

Velikost displeje - 480 x 800 pixelu
Počet barev – 65 536
Operační systém - Microsoft Windows Mobile 6.1 Professional
Procesor - Samsung S3C6410 800MHz
Grafický akcelerátor - Hardwarový
Operační paměť - 256 MB RAM
Přídavná paměť – 512 MB ROM
GPS přijímač – ANO (A-GPS)

DirectX možnosti:

Maximální počet vertexů – 32 768
Maximální velikost textury – 2048x2048 pixelu
Počet aktivních světel – 1024
Hardwarová podpora rasterizace – NE
Hardwarová podpora transformací a osvětlení – NE
Maximální počet backBuffer – 1

Parametry zařízení, které ovlivňují výpočetní výkon (procesor a paměť) jsou na dobré úrovní. Zařízení nepoužívá driver pro DirectX od společnosti Microsoft. Použitý driver se jmenuje D3DM BVS Driver a neumožňuje hardwarovou akceleraci. Zařízení, která podporují hardwarovou akceleraci DirectX jsou spíše ojedinělá.

9.2. Test výkonu

I když je na testovaném zařízení využíváno pouze softwarové akcelerace, běží aplikace velice svižně. Nezaznamenal jsem žádné problémy při vykreslování v podobě sekání nebo trhaní obrazu při posuvech a rotacích.

Zkušební test aplikace proběhl také pomocí emulátoru, který je součástí Windows Mobile 6 SDK. Všechny částí aplikace fungují správně. Problém při použití emulátoru je hlavně při zobrazování 3D scény pomocí DirectX Mobile. Dochází k výraznému sekání obrazovky. Proto je při vývoji aplikace nutné využívat fyzického zařízení, na kterém se dají testy provádět.

9.3. Test v terénu

Aplikace byla otestována na lokalitě Ostravské ZOO. Testování proběhlo za dobrého počasí. Průměrný počet satelitů ve výhledu se pohyboval okolo 5. Při chůzi vykazovala navigace odchylky v řádu několika metrů. K zpřesnění došlo až při zastavení a stání se zařízením na jednom místě v řádu desítek sekund.
GPS informace byly k dispozici z několika zdrojů[41]. Informace o jednotlivých pozicích se lišily a jako nejpřesnější se ukázaly informace převzaté z katastrálního úřadu. Jako příklad je zde uvedena tabulka 9.3 s porovnání GPS hodnot z jednotlivých zdrojů. Všechno hodnoty byly převedeny do systému S-JTSK, aby je bylo možné porovnat.

Zdroje dat	**Body na mapě**							
	1		**2**		**3**		**4**	
	X	**Y**	**X**	**Y**	**X**	**Y**	**X**	**Y**
OpenStreetMap	1100442	467199	1100570	467268	1100425	467270	1100409	467197
Google Maps	1100410	467196	1100520	467221	1100411	467261	1100353	467211
ČKÚ*	1100410	467193	1100519	467225	1100411	467263	1100353	467211
Naměreno	1100408	467193	1100517	467224	1100412	467264	1100353	467214

* Český katastrální úřad

Tab. 9.3 – Porovnání souřadnic

[41] Použité zdroje dat viz. kapitola 7.1 Zdroje GPS Dat

Závěr

Podle zadání diplomové práce se mi povedlo vytvořit vzorový 3D model a aplikaci, která tento model zobrazuje. Vytvořený 3D model slouží pouze jako demonstrace, jak lze při tvorbě takového modelu postupovat a co je třeba vzít v úvahu. Aplikace umožňuje vyžití navigace za pomoci vestavěného GPS přijímače a navíc umožňuje import mapy a jejích informací z XML souboru. Aplikaci tak může využít kdokoli s vlastní vytvořenou mapou.

Knihovna .NET Compact Framework poskytuje velice dobré zázemí pro programování aplikací jakéhokoli druhu. Pravdou je, že kompaktní implementace postrádá některé prvky, které mohly být z desktopové verze knihovny implementovány. Jejich nepřítomnost se však dá vyřešit použitím jiných prvků nebo napsáním vlastních tříd.

Knihovna DirectX Mobile umožňuje tvorbu 3D scény odpovídající restrikcím mobilních zařízení. Jedinou funkci, kterou jsem postrádal je hitTest, bez něj není možné zjišťovat, na kterou budovu (objekt ve 3D) uživatel klikl. Aplikace tudíž postrádá funkci, kdyby se po kliknutí na budovu zobrazily její podrobné informace.

Mezi problémy s vývojem software pro mobilní zařízení patří přenositelnost aplikace. Mnou vytvořená aplikace je omezena pouze na platformu Windows Mobile 6 až 6.5. Postup při tvorbě a struktura výsledné aplikace však může posloužit k vývoji aplikací i pro jiné platformy.

Literatura a zdroje informací

[1] J. Gumster: *Blender For Dummies*. Indiana, Wiley Publishing, 2009, ISBN: 978-0-470-40018-0.

[2] W. F. Engel: *Beginning Direct3D Game Programming*. Boston , Premier Press, 2003, ISBN: 1-931841-39-X

[3] R. Penton: *Beginning C# Game Programming*. Boston , Thomson Course Technology, 2005, ISBN: 1-59200-517-9.

[4] W. Jones: *Beginning DirectX 9*. Boston , Premier Press, 2004, ISBN: 1-59200-349-4.

[5] R. Dostál: *Úvod do GIS*. Přednáška ke cvičení, Ostrava , VŠB-TU HGF, 2004.

[6] *DirectX Tutorial* [online].
Dostupné na URL: http://www.directxtutorial.com/index.aspx (květen 2010).

[7] *DirectX using c#* [online].
Dostupné na URL: http://www.riemers.net/eng/Tutorials/dxcsharp.php (květen 2010).

[8] *Mobile DirectX with the Compact Framework* [online].
Dostupné na URL: http://www.mperfect.net/cfMDX/ (květen 2010).

[9] *Direct3D Tutorials* [online].
Dostupné na URL: http://www.drunkenhyena.com/cgi-bin/directx.pl (květen 2010).

[10] *Blender Tutorials* [online].
Dostupné na URL: http://www.nystic.com/ (květen 2010).

[11] *Projekce Mapy* [online].
Dostupné na URL: http://wikipedia.infostar.cz/m/ma/map_projection.html (květen 2010).

[12] *Blender 3D: Noob to Pro/UV Map Basics* [online].
Dostupné na URL: http://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro/UV_Map_Basics (květen 2010).

Příloha A:

Výpis z SVG souboru s cestami ZOO

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg id="svg9654" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"        xmlns="http://www.w3.org/2000/svg"        height="2000pt"
width="2000pt"                                          version="1.1"
xmlns:cc="http://creativecommons.org/ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/" viewBox="0 0 1945 1983">

<path     fill="none"     stroke="blue"     stroke-width="1"     d="M
38.8877851326717 48.8881994027353 L 40.4402352823643 50.1619725951983
L     42.4133933268022     51.8824127731612     L     44.1992630701046
53.5081983159529  "/>

<metadata id="metadata14269">
<rdf:RDF>
<cc:Work rdf:about="">
<dc:title/>
</cc:Work>
</rdf:RDF>
</metadata>
</svg>
```

Příloha B:

Textura modelu

Příloha C:

Vzorový vstupní XML soubor

```
<?xml version="1.0"?>
<MapData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <MapName>ZOO Ostrava</MapName>
  <MapScale>10</MapScale>
  <MapZeroPoint X="1100408" Y="467195" />
  <MapLayersFiles>
    <string>land.md3dm</string>
    <string>roads.md3dm</string>
    <string>buildings.md3dm</string>
  </MapLayersFiles>
  <MapFolder>Mesh\</MapFolder>
  <AttractionFolder>Attractions</AttractionFolder>
  <FieldOfViewMin>30</FieldOfViewMin>
  <FieldOfViewMax>600</FieldOfViewMax>
  <FieldOfViewStart>300</FieldOfViewStart>
  <Attractions>
    <Attraction Name="Pavilon hrochu a nosorozcu">
      <Location SJTSKx="1100518" SJTSKy="467264" />
      <SourceFile>pavilonNosorozcu.html</SourceFile>
    </Attraction>
    <Attraction Name="Pavilon Slonu">
      <Location SJTSKx="1100491" SJTSKy="467258" />
      <SourceFile>pavilonSlonu.html</SourceFile>
    </Attraction>
    <Attraction Name="Letni expozice lemuru">
      <Location SJTSKx="1100469" SJTSKy="467162" />
      <SourceFile>pavilonLemuri.html</SourceFile>
    </Attraction>
    <Attraction Name="Obcerstveni">
      <Location SJTSKx="1100404" SJTSKy="467193" />
    </Attraction>
  </Attractions>
</MapData>
```

Příloha D:

Příkladový detail atrakce - HTML kód

```
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <link rel="stylesheet" type="text/css" href="./style.css" />
</head>
<body scroll="no">
        <div id="content">
                <h1>Letní expozice lemurů</h1>

<p>Dva ostrovy naproti výběhu afrických zvířat obývají madagaskarské poloopice - <strong>lemuři korunkatí</strong> (<em>Eulemur coronatus</em>) a <strong>lemuři červenobřiší</strong> (<em>Eulemur rubriventer</em>).</p>
<p><em>Víte, že...<br />
    ... ačkoliv lemuři obývají pouze Madagaskar (a některé přilehlé ostrovy, kam byli vysazeni člověkem), jsou nejrozmanitější skupinou primátů na Zemi (i dnes jsou stále objevovány nové druhy). Protože obývají pouze omezené území, jsou zároveň jedněmi z <strong>nejohroženějších</strong> - jejich přirozený domov - deštný prales je kácen a odlesňován pro zemědělské účely.
</em></p>
<p>... název lemur pochází z lemuros, což bylo v antickém Římě označení pro duchy zemřelých, ozývajících se za nocí silným křikem. </p>

<div class="picture_box"><img src="lemur.jpg"><br />Lemur červenobřichý</div>

        </div>
</body>
</html>
```

CSS styly

```
body {
  margin:0px;
  background-color: lightyellow;
}

#content{
  width:480px;
  height:748px;
  overflow:auto;
```

```
  padding:5px;
}

h1{
  font-size: 16pt;
  text-align: center;
  border-bottom: 1px solid gray;
  letter-spacing: 2px;
}

p{
  /*text-indent: 20px;*/
}

.picture_in_text{
  float:left;
  padding: 0px 20px 0px 0px;
  margin: 0px 20px 0px 0px;
  border-right: 1px solid gray;
}

.picture_box{
  float:left;
  padding: 20px;
  text-align: center;
}
```

Příloha E:

Obsah přiloženého CD:

1. Text diplomové práce ve formátu PDF, soubor:
   Pavel Dvouletý – Mobilní navigace v prostředí DirectX - Diplomová práce.pdf
2. Text diplomové práce ve formátu DOC, soubor:
   Pavel Dvouletý – Mobilní navigace v prostředí DirectX - Diplomová práce.doc
3. Zdrojové kódy aplikace, složka: Application\EventGPS
4. Instalace aplikace CAB, soubor: Application\EventGPSInstal.cab
5. Vzorová data (XML), složka: Application\SampleData
6. Program OSMParser, složka: Application\OsmParser
7. Data z OpenStreetMap (SVG), složka: Svg files
8. Modely a textury, složka: Maps and models
9. Abstrakt a klíčová slova diplomové práce v češtině a angličtině, soubory:
   abstrakt a klíčová slova.txt
   abstract & keywords.txt